applied sciences

MDPI

Article

Towards a System Dynamics Framework for Human–Machine Learning Decisions: A Case Study of New York Citi Bike

Ganesh Sankaran [1], Marco A. Palomino [2,*], Martin Knahl [3] and Guido Siestrup [3,*]

[1] School of Engineering, Computing and Mathematics, University of Plymouth, Plymouth PL4 8AA, UK; ganesh.sankaran@plymouth.ac.uk
[2] School of Natural and Computing Sciences, University of Aberdeen, Aberdeen AB24 3UE, UK
[3] Faculty of Business Information Systems, Hochschule Furtwangen University, 78120 Furtwangen, Germany; knahl@hs-furtwangen.de
* Correspondence: marco.palomino@abdn.ac.uk (M.A.P.); guido.siestrup@hs-furtwangen.de (G.S.)

Abstract: The growing number of algorithmic decision-making environments, which blend machine and bounded human rationality, strengthen the need for a holistic performance assessment of such systems. Indeed, this combination amplifies the risk of local rationality, necessitating a robust evaluation framework. We propose a novel simulation-based model to quantify algorithmic interventions within organisational contexts, combining causal modelling and data science algorithms. To test our framework's viability, we present a case study based on a bike-share system focusing on inventory balancing through crowdsourced user actions. Utilising New York's Citi Bike service data, we highlight the frequent misalignment between incentives and their necessity. Our model examines the interaction dynamics between user and service provider rule-driven responses and algorithms predicting flow rates. This examination demonstrates why understanding these dynamics is essential for devising effective incentive policies. The study showcases how sophisticated machine learning models, with the ability to forecast underlying market demands unconstrained by historical supply issues, can cause imbalances that induce user behaviour, potentially spoiling plans without timely interventions. Our approach allows problems to surface during the design phase, potentially avoiding costly deployment errors in the joint performance of human and AI decision-makers.

Keywords: machine learning; system dynamics; simulation modelling; algorithmic decision-making; supply chain planning; New York City Citi Bike

Citation: Sankaran, G.; Palomino, M.A.; Knahl, M.; Siestrup, G. Towards a System Dynamics Framework for Human–Machine Learning Decisions: A Case Study of New York Citi Bike. *Appl. Sci.* **2024**, *14*, 10647. https://doi.org/10.3390/app142210647

Academic Editors: Tomislav Stipančić and Duska Rosenberg

Received: 22 October 2024
Revised: 10 November 2024
Accepted: 14 November 2024
Published: 18 November 2024

1. Introduction

Despite the rapid advancement of Artificial Intelligence (AI), particularly with generative AI [1], human judgment remains critical in decision-making [2–5].

From a macroeconomic perspective, AI, as a General-Purpose Technology (GPT), requires complementary structures, processes and systems to maximise its value [6,7]. This transition is gradual, particularly for organisations not born digital, which will blend old and new processes for some time [8]. While automation replicates existing rules, an augmentation approach—combining humans and AI—creates opportunities to innovate and extend task boundaries, effectively "growing the pie" by expanding the range of available options [9,10].

From an organisational decision-making perspective, AI's handling of tasks involving tacit knowledge is often unreliable due to the variability in such tasks [11–13]. Effective AI integration requires a shift from "point solutions" to a system-level approach that considers interdependencies within the organisation [8]. This transition from narrow "digitisation" to a holistic, problem-led "digital" view acknowledges the contextual richness of organisational decision-making, recognising the limitations of reducing complex problems to a few variables [14–16]. As Gigerenzer points out, simplifications are inadequate in a world that

lacks stability, necessitating a systemic approach that accounts for both uncertainty and the complexity of real-world contexts [17].

Technologically, the limitations of Large Language Models (LLMs) and the biases inherent in machine learning highlight that data and models are not entirely objective [18–20]. Machine learning models inevitably reflect the biases of their designers, refuting claims of "theory-free" AI [21,22].

These mutually reinforcing perspectives establish human–AI collaboration as the default for complex problem-solving. Research shows that coordination dynamics, already complex with traditional technologies and likened to navigating a rugged landscape, become even more intricate with the introduction of machine learning [23–25].

In light of the need for collaborative decision-making between humans and AI, the challenge is to make the dynamics of joint decisions transparent. Our proposed framework addresses this by integrating deep learning as the centrepiece of the algorithmic component, complemented by human judgment, within a simulation workbench designed to evaluate and refine decision policies. To demonstrate the framework's application, we applied it to inventory balancing in docked bike-sharing systems, which often experience spatial and temporal inventory asymmetries, where stations with similar initial stocks diverge over time. Two main motivating factors for choosing this problem were the availability of high-quality, publicly accessible datasets that Citi Bike regularly publishes [26] and its use of a crowdsourcing model for inventory balancing [27]. While recruiting users for inventory balancing instead of using carrier vehicles offers environmental benefits, the need to account for user behaviours in response to bike saturation at stations and incentives adds to the complexity of the coordination problem in a docked model. Our framework tackles this challenge by integrating heuristic and data-driven methods, enabling a comprehensive assessment of their performance across different experimental conditions.

From a systems perspective, organisations with hierarchical structures, interacting parts, beliefs, rules and goals can be viewed as complex systems. System dynamics, which focuses on dynamic complexity arising from interactions over time rather than the number of components, offers a viable method for understanding organisational behaviour [28,29].

In developing system dynamics, Forrester's early work examined fluctuations in performance factors within a manufacturing supply chain under stable customer demand patterns [30]—the so-called "bullwhip" effect, or amplification of demand variability up the supply chain. He called the general behaviour patterns in this context "industrial dynamics". Forrester then applied the system dynamics method to policy matters at the city level ("urban dynamics") and even on a global scale ("world dynamics") [31].

Although system dynamics has shown itself to be adept at modelling problems at different scales since its inception, traditionally, the focus has been on broad societal issues more than organisational topics [32]. Moreover, even when applied in organisational contexts, system dynamics models traditionally rely on heuristics to represent the mental model guiding decisions (e.g., [33–35]).

Despite the introduction of the Python library PySD, which allows combining Python's rich data science methods and simpler judgmental rules in system dynamics models [36], AI use has been restricted mainly to policy optimisation [37] and searching the space of inputs for a specific mode of behaviour [38].

Since one of the core principles of a complex system is its hierarchical structure—what Simon calls "boxes-within-boxes" [39]—characterised by more interactions within subsystems and fewer between them, there is an untapped potential for modelling organisational behaviour as a mix of relatively closed algorithmic subsystems and more open rule-based ones cohabiting, using PySD [40]. While traditional inventory balancing is often modelled as an "optimisation" problem involving carrier vehicles (e.g., [41–44]), incorporating user engagement complicates the planning problem and justifies a combination of heuristic and algorithmic approaches. As shown in Figure 1, modern organisations blend rule-based and AI-driven models, mirroring the heuristic–data science approach used in our bike-sharing study. However, despite integrating human judgment and algorithmic decision-making,

organisations often fail to evaluate the overall performance of joint decision-making comprehensively. This gap underscores the novelty of our model in facilitating double-loop learning [45], allowing real-world feedback to iteratively refine both heuristic and AI elements of the mental model, fostering more adaptive human–AI collaboration.

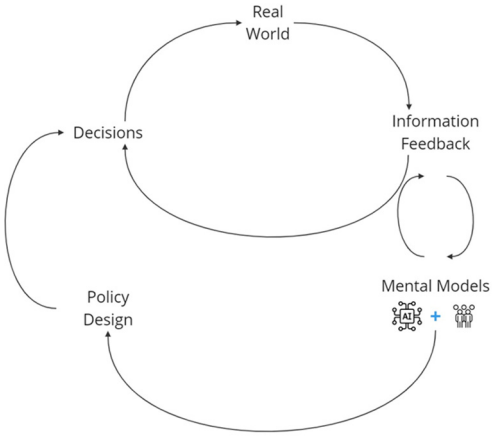


Figure 1. Double-loop learning applied to refine collaborative human–AI decision-making.

The remainder of this paper is structured as follows: Section 2 outlines the modelling framework, focusing on Phase 2 of the decision-making process, which encompasses evaluation, highlighting the integration of machine learning and heuristic methods. Section 3 applies this framework to the bike-share case, starting with the business context and problem illustration and progressing through data sources, causal mapping for evaluation, and the development of simulation and ML models. This section includes detailed subsections on demand forecasting, causal inference and stocks and flows modelling, leading into partial model testing and comprehensive integration analysis, such as assessing starting inventory policies and mapping the performance across the decision landscape. Insights and implications derived from these evaluations are then discussed. Finally, Section 4 summarises the key findings, contributions and potential directions for future research.

2. Modelling Framework

Our primary deliverable is a quantitative model designed to bridge the gap between recognising the need for machine learning (ML) in decision-making and implementing it. We define a "quantitative" model following [46], where causal relationships between variables are defined and quantified. The aim is to shift from qualitative conceptual models that suggest how joint human–ML decision-making might be structured to quantitative models that evaluate whether existing organisational policies and processes are conducive to successful ML implementation.

This approach follows established guidelines for system dynamics modelling [45,47], emphasising the need for a holistic understanding of problem-solving [48]. The framework operates within the evaluation phase of decision-making, which bridges conceptualisation and implementation. It leverages systems thinking principles, where the structure of the system determines its behaviour [28].

As depicted in Figure 2, the evaluation phase comprises four main steps: Causal Mapping, Formulation of Simulation and Machine Learning Models, Partial Model Testing and Integration and Policy Analysis. Each step plays a crucial role in transitioning from conceptual understanding to an actionable and testable model.

Causal Mapping is the foundation, representing current and future decision-making states through causal diagrams. This step abstracts operational details to define system boundaries, identify critical feedback structures, and set the stage for more detailed model formulation. The purpose is to make explicit the interdependencies and feedback loops within the system, ensuring that critical elements influencing decision outcomes are accounted for.

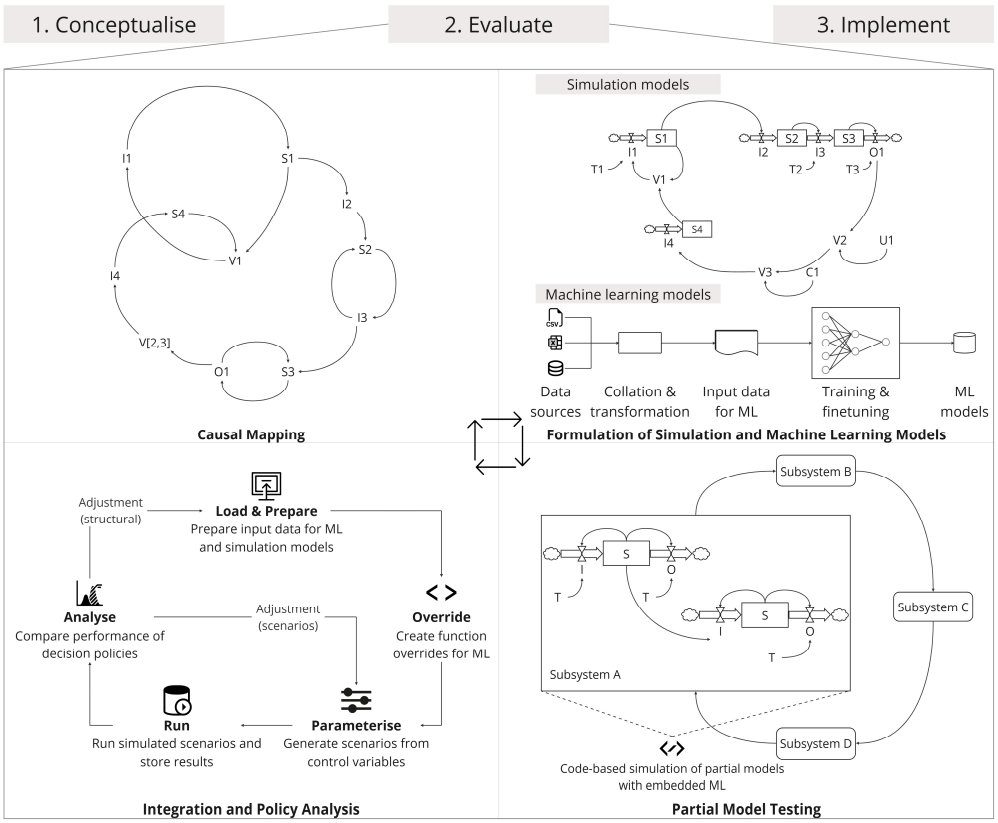


Figure 2. Overview of the modelling framework.

Formulation of Simulation and Machine Learning Models follows, employing tools such as Vensim, PySD and ML techniques (e.g., Recurrent Neural Network (RNN)). Here, causal maps guide the development of simulation models that integrate both ML and heuristic methods. The simulation models are structured to allow interaction with ML submodels, facilitating the exchange of information between predictive algorithms and rule-based processes. Data collation and transformation, shown in the diagram, prepare input data for ML training and testing, while simulation elements are parametrised to create and test various scenarios using different configurations of control variables.

Partial Model Testing is crucial for verifying the local rationality of subsystems, ensuring alignment with process-level objectives before full integration. In this phase, code-based simulations with embedded ML models are run to assess the performance of individual subsystems, such as those depicted as Subsystems A, B, C and D. This iterative testing step helps identify potential biases or misalignments within isolated components, allowing for adjustments before the entire model is synthesised.

Integration and Policy Analysis assesses the global rationality of the intervention by simulating system-wide outcomes and comparing them against organisational goals. This step explores potential misalignments and examines the robustness of different policy choices, informing adjustments needed for improved performance. The final output of this phase serves as a low-risk tool for evaluating whether to proceed with or adapt the human-AI strategy in a real-world implementation.

In summary, this modelling framework leverages the strengths of both simulation and ML to create a cohesive system that bridges conceptualisation and implementation. Using tools like PySD for programmatic simulation and incorporating a combination of heuristic and algorithmic approaches, the framework facilitates double-loop learning; this ensures that real-world feedback informs and adapts both heuristic and ML components, fostering a more adaptive human–AI collaboration.

3. The Bike-Share Case

3.1. Business Context and Problem Illustration

Docked bike-sharing systems, such as New York's Citi Bike [26], require customers to rent and return bikes at designated stations. The fixed nature of docked systems, compared to the flexibility of dockless models [49], can contribute to inventory imbalances that lead to outages, even when the overall bike supply is sufficient [50]. These outages—where bikes or docking slots are unavailable—limit both rentals and returns, potentially resulting in future stockouts. To manage these imbalances, providers may: (a) adjust starting inventory through overnight rebalancing, (b) monitor and rebalance during the day or (c) incentivise users to help redistribute inventory. While overnight adjustments (a) set a solid starting point, the variability of station demand can create imbalances throughout the day. Daytime rebalancing (b) is effective but may lead to negative externalities, such as vehicle emissions. Crowdsourcing (c), exemplified by Citi Bike's Bike Angels program, is appealing for its user engagement and environmental benefits [27]. Despite the potential of crowdsourced rebalancing, improper timing can limit its effectiveness.

Our analysis of hourly bike utilisation at seven stations within a selected demand cluster (criteria detailed in Section 3.3) over a typical week shows that, while individual stations reach high utilisation (calculated as Docks Available/Capacity Available, with high utilisation defined at an 85% threshold), simultaneous stockout risks across multiple stations are infrequent; this indicates opportunities for balancing inventory among nearby stations to avoid stockouts. To illustrate this timing issue, consider station E 20 St and Park Ave, which ranked second in the number of return incentives offered during August 2023 (these return incentives are offered to users to "give" or return bikes to the station in question.) We defined incentives as reasonable if offered above 70% utilisation, missed if not initiated above 85%, and excessive if given below 70%. Figure 3, highlighting correctly timed incentives in green and errors in red, shows that misaligned incentives were more frequent than effective ones. Given the potential effectiveness of crowdsourcing strategies, we hypothesise that better-timed incentives could have helped prevent some stockouts.

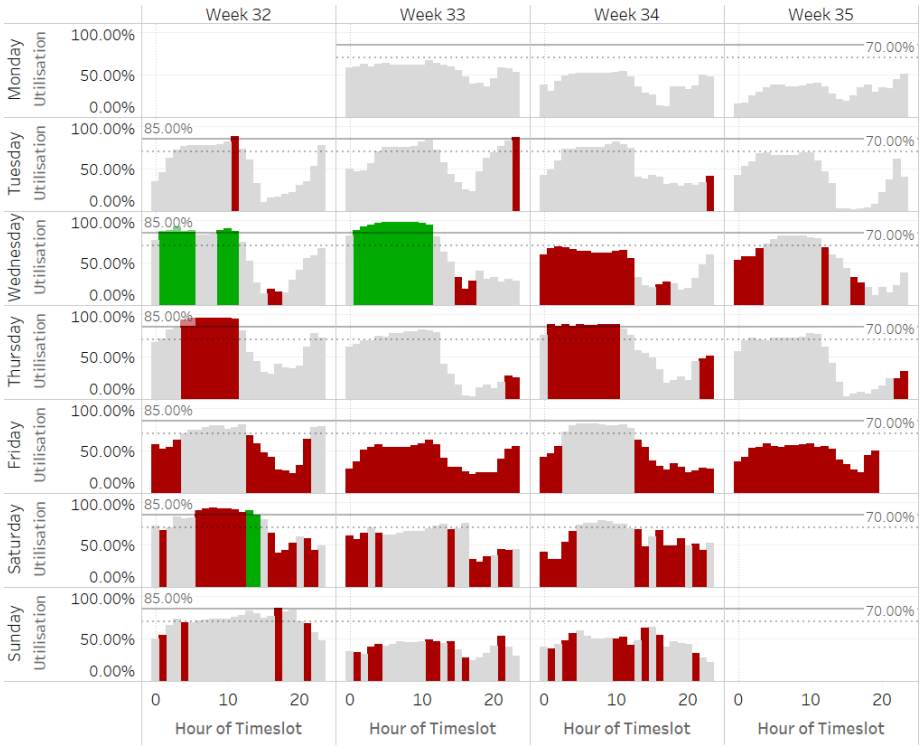


Figure 3. Analysis of suboptimal incentive timing at a Citi Bike station during August 2023, showcasing opportunities for improved inventory management through better-aligned incentives.

Conditional colouring formula:

COLOUR = {GREEN: if CU ≥ HUT and RI > 0; RED: if CU ≥ HUT and (RI = 0 or RI is null); RED: if CU < RHUT and RI > 0; GRAY otherwise}

where CU = Average Capacity Utilisation = AVERAGE(DA/CA), RI = Sum of Return Incentives, HUT = High Utilisation Threshold, RHUT = Reasonably High Utilisation Threshold, DA = Docks Available, and CA = Capacity Available.

3.2. Data

Forecasting user demand for rentals and returns is the first step in allocating bike inventories to stations and anticipating balancing needs. We base our forecasts on Citi Bike's monthly trip histories published online (https://ride.citibikenyc.com/system-data (accessed on 10 October 2024)).

The Bike Angels program incentivises users to balance inventory by offering redeemable ride points. Incentives for users to rebalance inventory are inherently local, as they redirect returns from overflowing stations to those in need, adjusting flows between nearby stations without significantly altering overall demand. Given the asymmetries in stockout patterns between nearby stations, modelling a carefully selected cluster of proximate stations is justified to test causal hypotheses.

We selected stations based on the following criteria: (a) availability of incentive data to correlate with demand, allowing analysis of its historical effectiveness using a causal inference model, which estimates the impact of incentives on user demand and adjusts demand accordingly, (b) at least one station in the cluster should rank high in demand or ride counts to ensure the analysis focuses on a high-impact cluster and (c) stations should be within walking distance, allowing users to switch between stations easily, thus balancing inventory through guided incentives to "give" at one station and "take" at another to improve overall balance. For a time, station incentive data was accessible via an API (https://layer.bicyclesharing.net/map/v1/nyc/stations (accessed on 10 October 2024)), but this ceased after 25 August 2023. Since these data are critical for analysis (criterion a), we focused on August 2023, the most recent month with available data. To meet criteria b and c, ride count at each station was used. E 17 St and Broadway, ranked fifth out of 400 stations, was chosen as the nodal station, with the additional requirement to have several nearby stations. We then selected stations within a 350-metre radius to form our demand cluster.

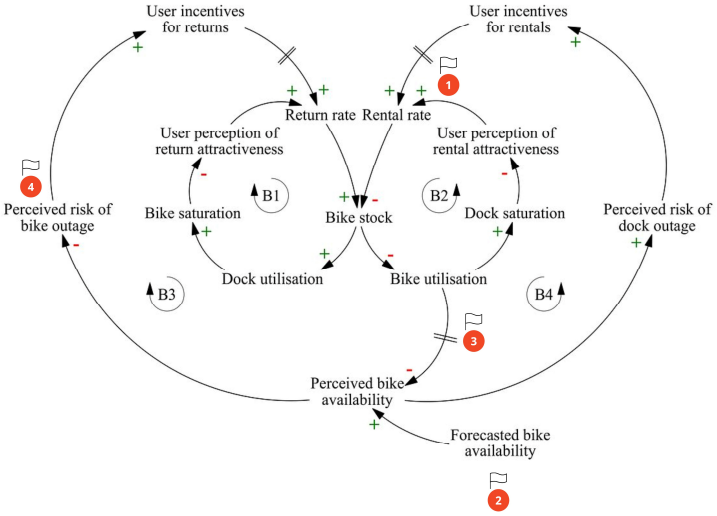
3.3. Evaluation—Causal Mapping

From the August 2023 data (Figure 3), we observe that, when a station's bike inventory is low (blocking rentals) or high (blocking returns), incentives are often either absent or incorrectly applied, indicating a flaw in the logic behind triggering incentives. In other words, since we have evidence that crowdsourcing as a policy is effective, we can conclude that improving the inventory balancing policy would reduce out-of-stock events (or no-service events). Given that the scope is inventory balancing, Figure 4 illustrates the hypothesised mental model, mapping the causal relationships between relevant variables, with an initial focus on a single bike station for clarity. There are three underlying sets of assumptions in the model: incentives influence user preferences for stations to rent from or return to, perceptions of bike saturation lead decision-makers to employ crowdsourcing to rebalance inventory and reduce stockouts and a station's attractiveness (nearly empty or full) prompts baulking behaviour. These assumptions are grounded in both intuition and empirical studies on Citi Bike and docked bike-sharing systems in general [27,51,52].

The *Return* and *Rental* rates represent the inflows and outflows of the *Bike stock*, with base rates reflecting demand influenced by weather or other contextual factors.

The difference between inflows and outflows, or the net inflow, integrates into the *Bike stock*. Specifically (note that s represents any point in time between t_0 and t):

$$Bikestock(t) = \int_{t0}^{t} [\text{Return rate}(s) - \text{Rental rate}(s)] ds + Stock(t_0).$$



Note on notation: A plus sign in the causal arrow that connects two variables (say, X and Y) says that an increase in X, all other things being equal, causes an increase in Y. A minus sign indicates that an increase in X, all other things being equal, causes a decrease in Y. The letter "B" in the illustration stands for "balancing" feedback loop. It is to show that the initial direction of change is opposed as one goes around the loop (the arrow indicates the direction of flow).

Figure 4. Causal Loop Diagram illustrating the feedback structures involved in bike-share inventory balancing and the hypothesised influences on system performance.

As bike stock increases, *Dock utilisation* (*Bike saturation)* at the station increases, assuming all else remains equal. This positive relationship (indicated by the plus sign on the arrow) reduces the station's attractiveness for returns (*User perception of return attractiveness* shown with negative polarity), causing users to avoid full stations and creating some baulking. The balancing loop "B1" resists this change, feeding back to stabilise the system.

A similar dynamic occurs in the balancing loop "B2". As the *Bike utilisation* or *Dock saturation* (number of free docks) increases, the attractiveness for rentals (*User perception of rental attractiveness*) decreases, reducing the rental rate and inducing stronger baulking at near-empty stations. The Formulation of Simulation and ML Models section will provide mathematical precision to these qualitative statements.

The outer loops "B3" and "B4" are more consequential to overall system behaviour. The *Perceived bike availability*, a smoothed composite of *Forecasted bike availability* and real-time *Bike utilisation*, influences the provider's perception of risk. Delays in incorporating real-time data may, however, occur, either due to system limitations or a deliberate "wait and see" approach (indicated by dashed arrows) where the provider does not want to react too soon to fluctuations (thus, "smoothing") [47]. As the perceived bike availability decreases, the *Perceived risk of bike outage* increases, which raises the pressure to incentivise returns (*User incentives for returns*). These incentives boost the return rate, increasing bike stock as intended. A similar dynamic plays out in the case of dock outage risk (*Perceived risk of dock outage*) that triggers more rentals (*User incentives for rentals*). It is assumed, both here and in the forthcoming equation formulation, that incentives reroute demand between stations rather than increase overall demand. When coordinated effectively, especially between nearby stations with asymmetric demand, crowdsourced inventory balancing through incentives reduces imbalances across the system.

In this simplified example, we observe multiple feedback loops with varying strengths, delays and non-linear relationships (e.g., the effect of bike availability on the provider's risk perception and user perception), making a closed-form solution unfeasible.

Based on the causal representation of the inventory imbalance problem, we might hypothesise about the potential root causes (see numbered flags in Figure 4). These are described below.

The baulking behaviour reduces recorded demand, effectively censoring flows (1). When used for forecasting, this results in a pessimistic outlook on future demand (2). In other words, historical issues hinder the ability to predict future demand accurately, worsening the situation over time. Additionally, inefficiencies in incorporating real-time availability or overemphasising it (where demand outliers or temporary fluctuations heavily influence decisions) (3) affect the provider's perception of stock levels. Lastly, the rules for triggering incentives, based on perceived and desired bike availability (4), impact their effectiveness.

The causal map's feedback-oriented view helps explain the dynamic behaviour behind inventory imbalance (the current state). Having identified the leverage points, we might further hypothesise a mental model that addresses the problem—the envisaged future state. Our proposed mental model likely involves a mix of human judgment and ML. For addressing bike-share inventory imbalance, several factors need to be considered: initiating incentives cannot be fully algorithmic due to budget constraints and other balancing methods; user response to station saturation is non-linear; it is better to base outage risk on patterns of actual and projected bike availability, rather than reacting to temporary demand fluctuations; and policy robustness is preferable over chasing "optimal" incentive schemes.

3.4. Evaluation—Formulation of Simulation and ML Models

Causal mapping establishes perceived bike availability, a synthesis of real-time and forecasted data, as the bike-share provider's key information source to assess outage risk and adjust incentive actions. Incorporating flow rate predictions helps smooth the impact of real-time fluctuations on outage risk perception. The data science component of the organisational mental model also includes predictions of how incentives will impact bike-share demand, specifically the expected increase in returns and rentals from crowdsourced inventory balancing. The accuracy of these predictions directly influences projected bike stock and future incentive decisions.

Figure 5 illustrates how these information flows—expected demand and response to incentives—impact material flows and system performance.

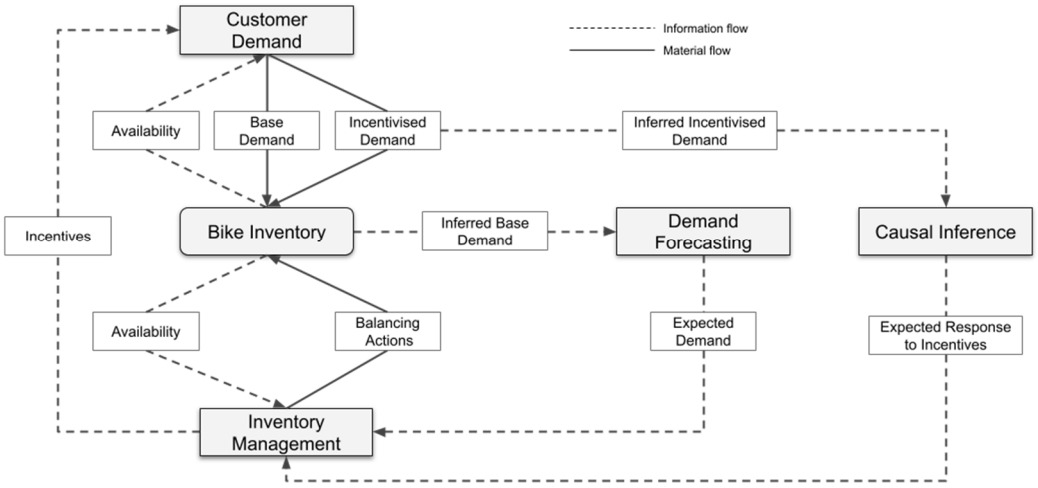


Figure 5. High-level overview of the bike-share inventory model, highlighting the main flows of rentals and returns and their interconnections within the system.

A standard metric for evaluating performance is the service level, which in the bike-share context can be defined as [51]:

$$\frac{(\text{potential customers} - \text{no-service events})}{\text{potential customers}}$$

In our case, improving prediction accuracy reduces no-service events, enhancing service levels. Bike-share demand, both rentals and returns (*Base Demand*), is driven by market demand, influenced by bike availability for rentals and dock availability for returns. Realised demand serves as the basis for *Demand Forecasting*. Through data cleansing (de-censoring), realised demand is adjusted to remove distortions from stockouts and outliers, producing a version closer to true market demand (*Inferred Base Demand*). The Demand Forecasting process then generates a forecast (*Expected Demand*), incorporating factors like weather forecasts that correlate with customer demand.

The *Inventory Management* process defines the provider's policies for ensuring adequate stock at individual stations (disaggregation of bike inventory) to meet expected demand. It also monitors real-time availability and triggers balancing actions or customer incentives to prevent stockouts. The process relies on expected demand and incentive impact predictions (*Expected Response to Incentives*) based on causal inference from historical incentive effects (*Inferred Incentivised Demand* from underlying *Incentivised Demand*).

We now discuss the ML forecasting process and causal inference that form the backbone of the algorithmic content in the mental model.

3.4.1. Demand Forecasting

Demand De-Censoring

A crucial step in preparing the historical rental and return data for forecasting is removing outliers or de-censoring demands. Outlier correction aims to eliminate the impact of extraordinary circumstances [53], providing a solid base for forecasting [54]. Unusual events disrupt the stationarity assumption, which is essential for reliable forecasting, as it assumes historical patterns can be extrapolated to predict future demand. Outlier correction seeks to minimise these disruptions.

For bike-share data, we detect and correct two types of outliers. The first occurs when the provider cannot meet rental or return demand due to insufficient bike or dock inventory, *censoring* legitimate demand and incorrectly assuming the supply issue will continue. The second type arises from inventory balancing actions, such as bulk returns or rentals, which deviate *demands* significantly from the average. Below, we outline the data preprocessing and cleansing steps used to create an outlier-corrected rentals and returns file.

Data preprocessing. Citi Bike's trip data contain transactional information used to determine bike-share rental and return demand. The key fields are location details and ride start and end times.

We used Tableau Prep Builder (https://help.tableau.com/current/prep/en-us/prep_about.htm (accessed on 10 October 2024)) to transform the source data into the desired format for demand forecasting. We applied a filter to the station fields (start or end) to focus the data on the nine stations selected for the case study's demand cluster. We split each day into ten-minute intervals and assigned ride start and end times to their respective slots. This aggregation reduces granularity, minimises demand variability, and improves forecast accuracy. Based on a preliminary analysis showing an average trip duration of 11 to 13 min, we selected ten-minute intervals to ensure we capture short trips to and from the same station. We then split rentals and returns into two streams for individual forecasting.

For the incentives data from Citi Bike's Bike Angel program, we developed a Python script to run every ten minutes and pull real-time data. Using the fields "name", "last_reported", and "bike_angels_action", we determined the timing of incentives at the selected stations. Figure 6 illustrates the data pipeline, from demand and incentive data to the two critical prediction types for inventory management.

With the *Raw Demands* and *Incentives Data* in place, we split the demands into *Incentivised Demands* and *Unincentivised Demands* (or *Base Demands*).

Unincentivised or Base Demands. If a demand period at a ten-minute time slot for individual stations has no incentive, the base demand equals the raw demand. If an incentive is applied, the base demand is the average demand for that timeslot at the station and filtered for periods without incentives during that month.

Incentivised Demands. If a demand period is under incentive, the incentivised demand equals the raw demand. Otherwise, it equals the average demand for that timeslot at the station/month, filtered for periods with incentives.

Outlier correction. A demand is considered an outlier if it exceeds the Upper Threshold Limit (UTL) or falls below the Lower Threshold Limit (LTL). For Base Demands, outliers are those below the 10th percentile or above the 95th percentile. For Promoted Demands, the LTL is the 25th percentile, and the UTL is the 99th percentile. Any detected outliers are corrected to the average value for the same timeslot and station, excluding outliers from the calculation.

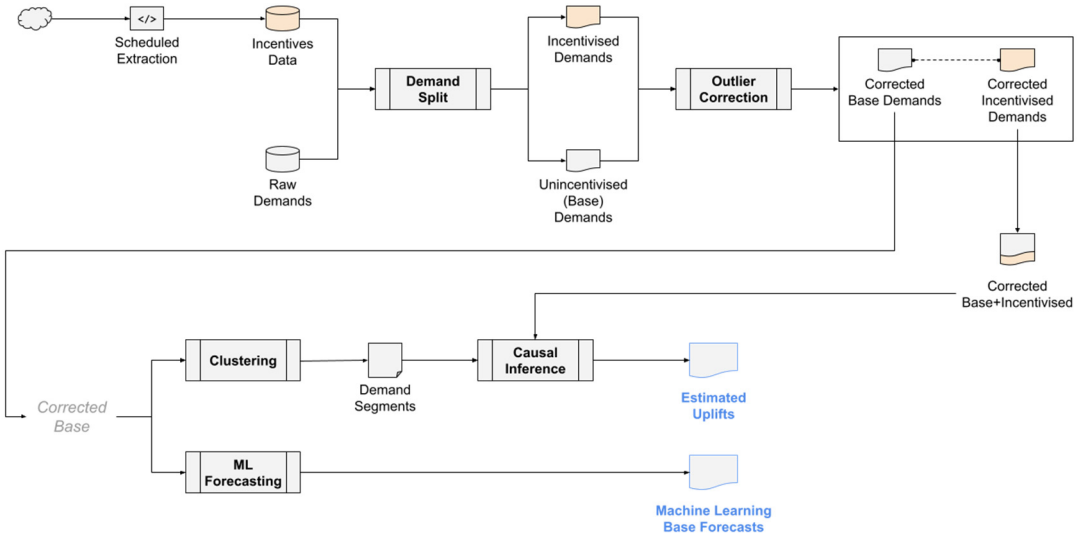


Figure 6. Data pipeline detailing the flow from raw demand and incentive data collection to the processing steps that generate critical input for forecasting models.

Machine Learning Forecast

Given that various contextual factors—such as temporal patterns, weather, local events and urban dynamics—affect bike-share demand, predicting it is a complex function of these inputs, making machine learning approaches like RNN ideal.

A crucial design decision in the RNN architecture is determining the type of inputs passed to the Dense output layer. The last recurrent layer can either return sequences, passing outputs across all timesteps, or return only the final timestep's output as a vector [55]. As we will explore later, this proved a significant leverage point for improving forecast accuracy in the bike-share case study.

Implementation details. The outlier-corrected base demands for rentals and returns served as the primary inputs for training the RNN on the bike-share data. We also downloaded historical weather data for August 2023 from OpenWeather (https://openweather.co.uk/about (accessed on 10 October 2024)) using location details around the nodal station (E 17 St and Broadway).

Besides weather, we also experimented with other correlates such as lagged demands (supplying demands with an n-period lag as additional features in the current timestep), demands of nearby stations with symmetric demands and temporal signals. Table A1 in the Appendix A details the data inputs, model architecture, parameters and features we tested and used for the final evaluation.

Weather data had a marginal impact on forecast accuracy, likely due to training on a single month's data. Similarly, nearby station demands and lagged demands showed no significant contribution. After several trials, we settled on using temporal signals and rental demands (to forecast returns and vice versa) as key features. By applying sine and cosine transformations on the time of day and day of the week, we created six features per prediction type (rental or return).

Traditional RNNs suffer from memory loss over long sequences due to repeated transformations, which leads to information degradation ([56], pp. 297–300). Given the granular nature of bike-share data (144 periods per day), we used Long Short-Term Memory (LSTM), a type of RNN, for the model architecture, which mitigates this by carrying information across timesteps. Acting like a "conveyor belt" ([56], pp. 297–300) that selectively retains or discards memory, LSTM improves accuracy by focusing on relevant data. Another key improvement came from training the network on every timestep (sequence-to-sequence) rather than only the final timestep.

A sequence-to-vector approach would output a tensor shaped as (samples, forecast horizon). In a sequence-to-sequence setup, in contrast, each timestep generates a tensor shaped as $(\text{samples}, \text{timesteps}, \text{forecas thorizon})$. This structure enhances performance, as omitting sequences can reduce accuracy by 40%. With predictions and losses at each timestep, more error gradients flow back, and shorter gradient travel distances (via *Backpropagation Through Time*) reduce information loss, further improving performance.

To evaluate the effectiveness of our chosen machine learning approach, we compared the LSTM model's performance with that of naive and exponential smoothing (EXS) methods. The LSTM model demonstrated significant improvements, with an average performance enhancement of approximately 26% for rental forecasts and 25% for return forecasts over EXS. These results underscore LSTM's superior ability to model complex temporal patterns and relationships, contributing to more accurate bike-share demand predictions.

3.4.2. Causal Inference

We apply a causal inference model proposed by Brodersen et al. using a Bayesian framework [57] to infer the impact of incentives on rental and return flows.

The model predicts the impact of an intervention (e.g., Bike Angel incentives) on the response metric (e.g., demand uplift) by regressing its pre-intervention time series on covariates unaffected by the intervention. It then generates a counterfactual response for the post-intervention period based on these "untreated" covariates. The impact is the difference between the observed and predicted counterfactual values.

To identify suitable contemporaneous correlates, we first compute similarity scores for station pairs in the demand cluster using the bike utilisation metric (as outlined below). For each station, we then select the time series of one or more of the closest stations.

Stage 1: Clustering by Demand Symmetry

Data: Hourly bike utilisation for each station (Bike utilisation (percentage of bikes rented) = Dock availability/Capacity).
Steps:

- Load and preprocess the data:
  - Pivot the dataset to create a 24 h utilisation vector for each station and day.
- Normalise the utilisation vectors for each station using MinMax scaling.
- For each pair of stations (21 pairs for seven stations):
  - Extract the 24 h utilisation vectors for corresponding days for both stations.
  - Calculate the L2 distance between the normalised vectors for each day.
  - Compute the average L2 distance across all days as the final similarity metric for the station pair.
- Store the average L2 distances for all station pairs in a table, sorted by similarity (lowest to highest distance).

In the second stage, we use the causal impact model to estimate the effects of incentives. The demand censoring step provides two continuous time streams for the unincentivised and incentivised demands. For the target station, we pass the incentivised demands for the post-intervention period to the model to infer what would have happened without the intervention. The inputs include the unincentivised demands from correlates, which are stations with similar demand patterns (identified in stage one).

Stage 2: Causal Impact Model for Estimating Incentive Effects

Data: Base (unincentivised) and incentivised rental/return demands for each station.
Steps:

- Prepare the Data:
  - Filter and rename the columns to distinguish between base and incentivised demands (e.g., "Rentals Normal" for base, "Rentals Uplift" for incentivised).
  - Pivot the data to create a time series of rentals and returns for each station, with pre-intervention values using base demand and post-intervention values using incentivised demand.
  - Define the pre- and post-intervention periods based on the intervention days.
- Run the Causal Impact Model:
  - Use the base demands from correlated stations (identified in Stage 1) as covariates and the incentivised demand for the target station as the response series.
  - The model infers the base demand for the target station during the post-intervention period (had no incentives been applied).
- Collate and Average Results:
  - Collect the target station's inferred rental/return uplift (i.e., the difference between the incentivised and predicted base demand) across all intervention days.
  - Average the results to create a typical week, showing hourly estimated percentage uplifts for rentals and returns.

The chart in Figure 7 compares the counterfactual (or inferred base) and incentivised rental demands for a given day in the test period. The orange band represents a 90% prediction interval around the point predictions for unincentivised demands. Given this interval, the causal inference model reported statistically significant effects across all test days, indicating that 90% of probable untreated demand values are expected to fall below the observed incentivised demands.

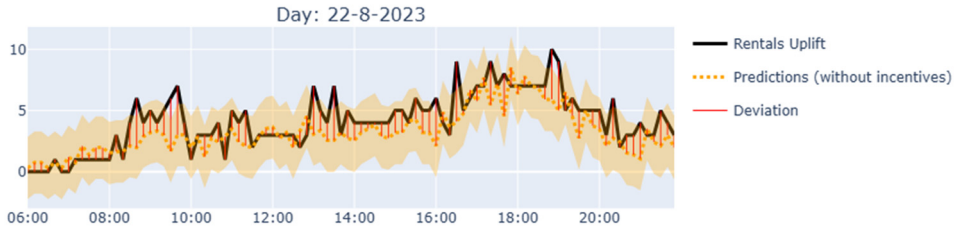


Figure 7. Causal impact analysis showing the effect of incentives on rental demand for one test day, illustrating key results.

3.4.3. Stocks and Flows Modelling

A common way to summarise systems thinking is by stating that a system's behaviour stems from its structure. When we unpack "structure", it consists of stocks, flows and feedback. However, in our causal mapping of problematic behaviour in the bike-share case and potential interventions, we did not explicitly distinguish between stocks and flows; instead, we focused on the interconnected feedback loops to offer plausible explanations. While identifying key variables and their relationships suffices to hypothesise behaviour patterns, precise quantification for simulation requires clearly distinguishing between stocks and flows.

The stocks and flows schematic in Figure 8 illustrates the kind of causal map elaboration needed to advance in the simulation process. It highlights the decision-making role of feedback structures and emphasises the importance of stock and flow variables.

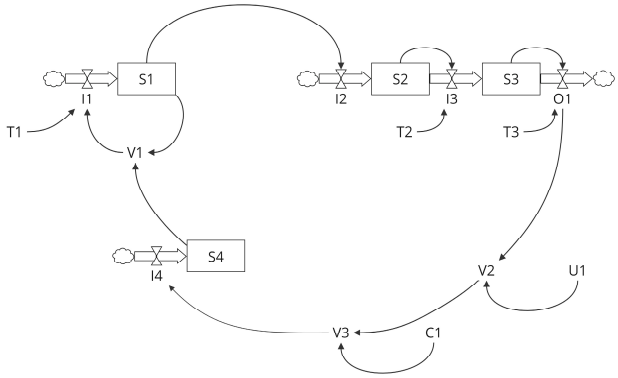


Figure 8. Diagram of the stocks and flows structure, demonstrating how key system elements interact in the simulation model.

The feedback or decision structures process cues from stock variables across different system sectors to make decisions that alter flow rates, which, in turn, feed back into the stocks. For example, the inflow rate (I1) into stock S1 depends on S1 and stock S4. Endogenous variables (V[1–3]), exogenous variables (U1), constants (C1) and time constants (T[1–3]) simplify the specification and maintenance of algebraic equations in simulation tools like Vensim. Essentially, net inflows are functions of stocks (with constants acting as slow-changing stocks) and exogenous variables; the intermediaries help break complex formulations into manageable parts.

This formulation provides the precision to quantify the interrelationships and determine the differential equations for net inflows. Once defined, the system's behaviour over time is simulated by numerically integrating these equations to evolve the stocks.

In the bike-share model, focusing on a single station, the bike stock represents the physical inventory that generates information cues for decision-making, influencing future flows. This stock reflects the real-world availability of bikes. A parallel structure also projects the future stock based on machine learning-predicted return and rental flows. The inventory management policy synthesises these two streams to assess outage risks, accounting for reporting and information delays. A set of rules translates this perception into decisions to incentivise returns or rentals.

A further algorithmic component that plays a crucial role in the feedback is the expected impact of incentives. During scenario simulations where various plausible rental and return flows operate on the real-world stock (our focus in the last two steps of the evaluation phase), the quality of predictions about incentive effectiveness influences our assessment of the effectiveness of policy variables. For example, if the accuracy of the causal impact of crowdsourcing inventory balancing through incentives is poor, the simulated real-world stocks will deviate from reality, skewing our assessment.

If we remove the single-station constraint, additional complexity arises from coordinating incentive actions by leveraging the complementarity between stations or clusters. Specifically, the inventory status of a partner station or cluster with asymmetric demands informs whether to apply incentives. The future-state inventory policy also incorporates a target range for bike availability or station utilisation. The distance between current perceived availability and the upper or lower threshold determines perceived risk, guiding stock levels within these limits.

We now review the two main components (in the corresponding stocks and flows formulation), synthesising information cues about the current and anticipated system states. For clarity, in Figure 9, we focus on the returns flow, noting that the dynamics are similar for rental flows. We also limit our review to a single-station model to simplify how

the formulation of stocks and flows enables the mathematical representation needed for simulations. (Figure A1 in the Appendix A provides the full structure, and the GitHub repository contains the equations for the two-station/cluster model.)

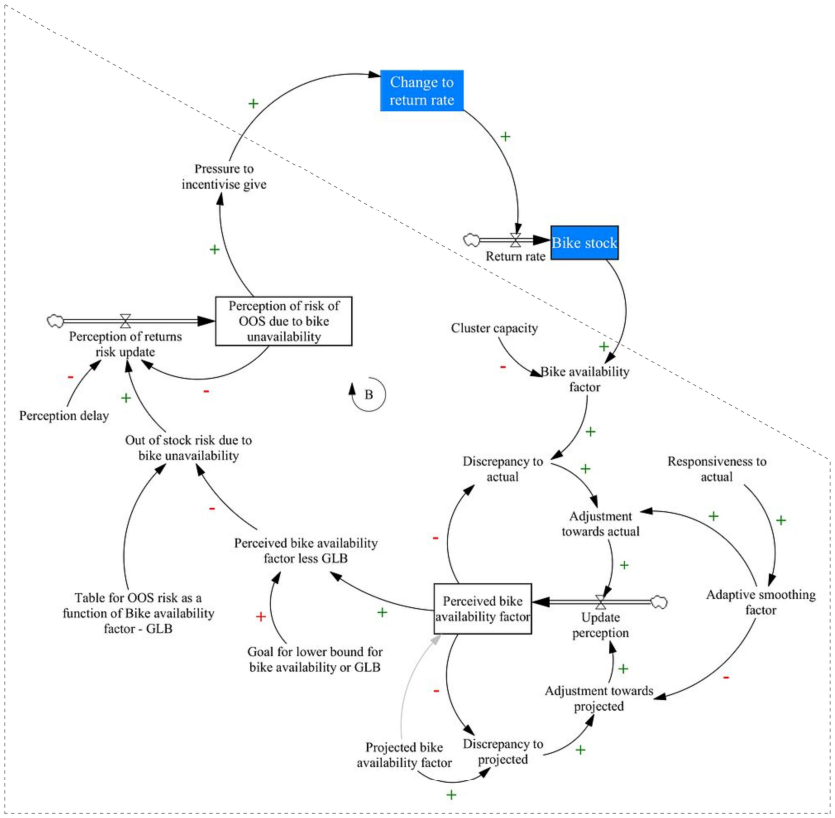


Figure 9. Focused view of a single station's stocks and flows structure, illustrating how return dynamics are modelled.

The first component is the formation of perception about availability, represented by the variable *Perceived bike availability factor* (PeBAF). This stock variable is shown in the diagram, which zooms in on the stocks, flows and feedback structure used to simulate the real-time behaviour of the system's return flows.

Since the decision-maker anchors their perception on forecasted availability, the real-time simulation connects to this critical information through the *Projected Bike Availability Factor* (PrBAF). To initialise PrBAF before the actual simulation, we run a model using forecasted demand flows (all done programmatically in a Python test workbench). Additionally, as the provider may update projections multiple times daily, a decision parameter controls the refresh frequency, which determines the look-ahead horizon. The process begins by aligning the initial projected stock with the actual stock at the start of the look-ahead horizon. The projected factors feeding into the real-time model are the ratio of projected stock to the station or cluster's maximum capacity.

With these definitions, we present the algebraic equations defining the variables involved in the decision-making process, ultimately shaping the perception of the station or cluster's inventory status.

1. Perceived bike availability factor (t) = Perceived bike availability factor $(t - dt)$ + Update perception $\times$ dt
2. Initial Perceived bike availability factor = Projected bike availability factor.
3. Update perception = Adjustment towards actual + Adjustment towards projected.
4. Adjustment towards actual = Adaptive smoothing factor $\times$ Discrepancy to actual.
5. Adaptive smoothing factor = f (time, Responsiveness to actual)

6. Discrepancy to actual = Bike availability factor − Perceived bike availability factor
7. Adjustment towards projected = (1 − Adaptive smoothing factor) × Discrepancy to projected.
8. Discrepancy to projected = Projected bike availability factor − Perceived bike availability factor.
9. Projected bike availability factor = g (rental and return demands, other factors).

In the structural equations, the delay variable *Responsiveness to Actual* determines how quickly the decision-maker integrates the actual Bike Availability Factor (BAF) into their perception. The *Adaptive Smoothing Factor* (ASF), a non-linear logistic function f of the current look-ahead step and responsiveness, weighs the perception update. This update combines the deviations from actual (BAF) and projected availability (PrBAF), with ASF applied to the actual deviation and (1 − ASF) to the projected deviation. The function g represents the generation of projections based on the machine learning forecast, which utilises historical demands and other features.

The second component is the risk perception, which gradually updates based on the *Perception delay* to reflect the current availability perception; this follows the classic anchor-and-adjust heuristic. In the case of risk, the stock variable retains some memory of prior risk, with the delay variable modelling the inertia. Figures 10 and 11 show the simulation result where the projected stock starts at 20 units (about 31% of the station's maximum capacity).

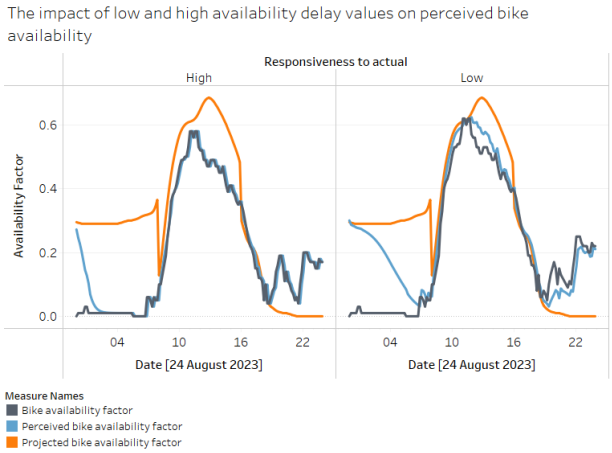


Figure 10. Comparative analysis of availability factor curves under two levels of responsiveness.

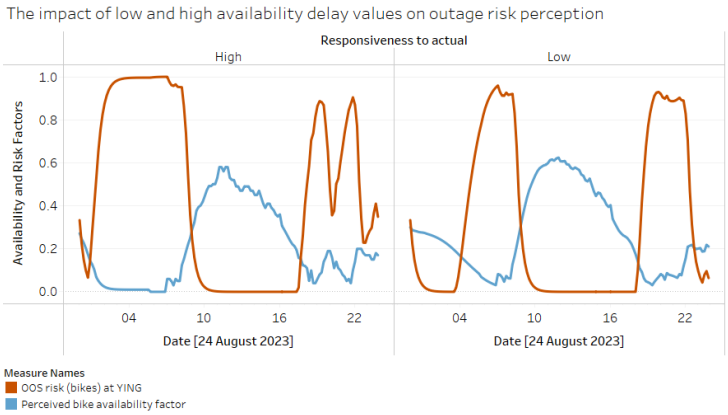


Figure 11. Comparative analysis of risk perception curves under two levels of responsiveness.

In contrast, the initial stock in the real-time model is set to zero, creating a significant discrepancy between BAF and PrBAF. The graphs display the availability and risk curves for high and low responsiveness levels. With a look-ahead horizon of 48 timesteps, we see

the projection align with the actual around 8 AM (six timesteps each hour) after the first refresh. A consequence of low responsiveness is that the risk perception curve is much narrower earlier in the day, as perception aligns more closely with the artificially inflated projected availability values.

3.5. Evaluation—Partial Model Testing

The objective of partial model testing, as suggested by Morecroft, is to broaden the perspective from the task level—typically the initial focus of the intervention—to encompass its immediate process context before fully integrating it into the holistic model. For more traditional policy changes that do not involve machine-learning agents, Morecroft advocated for partial testing to prevent what he called the "leap of logic" from equations to consequences [58].

This approach assumes that, under bounded rationality, policy interventions are generally reasonable at the local or subsystem level before accounting for the dynamic complexity of the entire system. Partial testing, therefore, helps confirm that an intervention is rational in intention before expanding its scope. Since machine learning introduces new dynamics, particularly in navigating decision-making alongside humans, partial testing becomes even more critical.

In the bike-share experiment, we apply this approach to test the reasonableness of using a deep learning method for demand forecasting instead of the traditional exponential smoothing technique—a well-established statistical procedure [59]. While the forecast accuracy metric showed substantial improvement with the RNN model, which intuitively makes sense given its ability to leverage correlates of demand that a univariate method like exponential smoothing cannot, we broaden the testing scope and use a process-oriented metric to determine if these improvements hold at this level.

The following steps outline the procedure for executing the partial tests.

Inputs:

ML (LSTM) and exponential smoothing (STAT) predictions for a given bike station/day.

Maximum capacity of a station.

Procedure:

- Calculation of Just-in-Time Inventory Balancing (Receipts and Issues):
  - Run the ML and STAT predictions through a simplified stocks and flows structure to compute projected stocks.
  - Whenever an outage occurs (either lack of bikes or docks), record the volume as a planned balancing receipt or issue.
- Calculation of the Process Metric: Total Outages with Just-in-Time Balancing (ML vs. STAT Predictions):
  - Run actual or simulated rental and return demands through a structure that models actual stock (aligned with the projected stock structure).
  - Use planned receipts and issues (from the previous step) as lookups to adjust rentals and returns.
  - Record the bikes and docks outages.

Figure 12 depicts the structure of the stocks and flows used to compute the process metric. Despite its simplicity, the partial test helps determine whether individual rationality—in this case, the superior forecast accuracy of the RNN's rental and return predictions—translates to collective rationality. The latter refers to whether the combined use of these predictions in a more realistic setting continues to outperform the simpler alternative.

In our first iteration, the performance of several stations under ML was unexpectedly worse. The charts in Figure 13 illustrate a typical outcome. The blue and red lines in the stock outages chart represent the process metric for the station Broadway & E 21 St from 22–25 August 2023, for STAT and ML, respectively. Despite ML's forecast accuracy (mea-

sured by Mean Squared Error) being about 28% better for rentals and 19% for returns, its process metric is 95% worse, with 72 units versus 37 units of combined bike and dock outages. A closer inspection, tracing the red and blue arrows, reveals the reason by comparing the projected and actual utilisation over time.

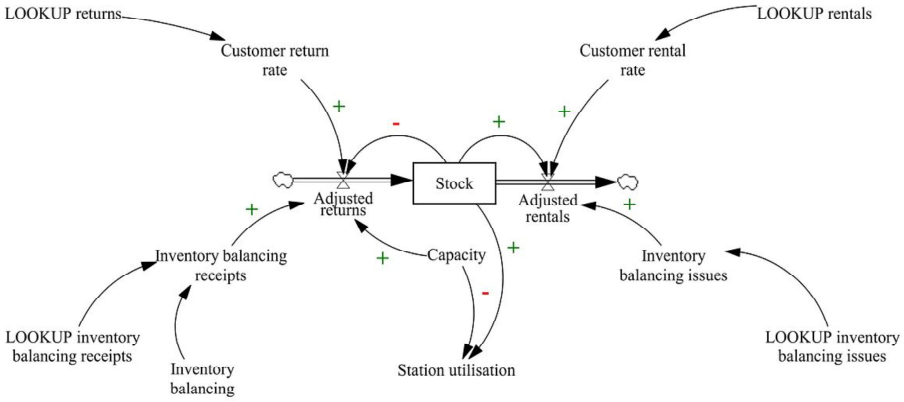


Figure 12. Simplified representation of the partial testing model used for verifying local rationality before full-scale integration.

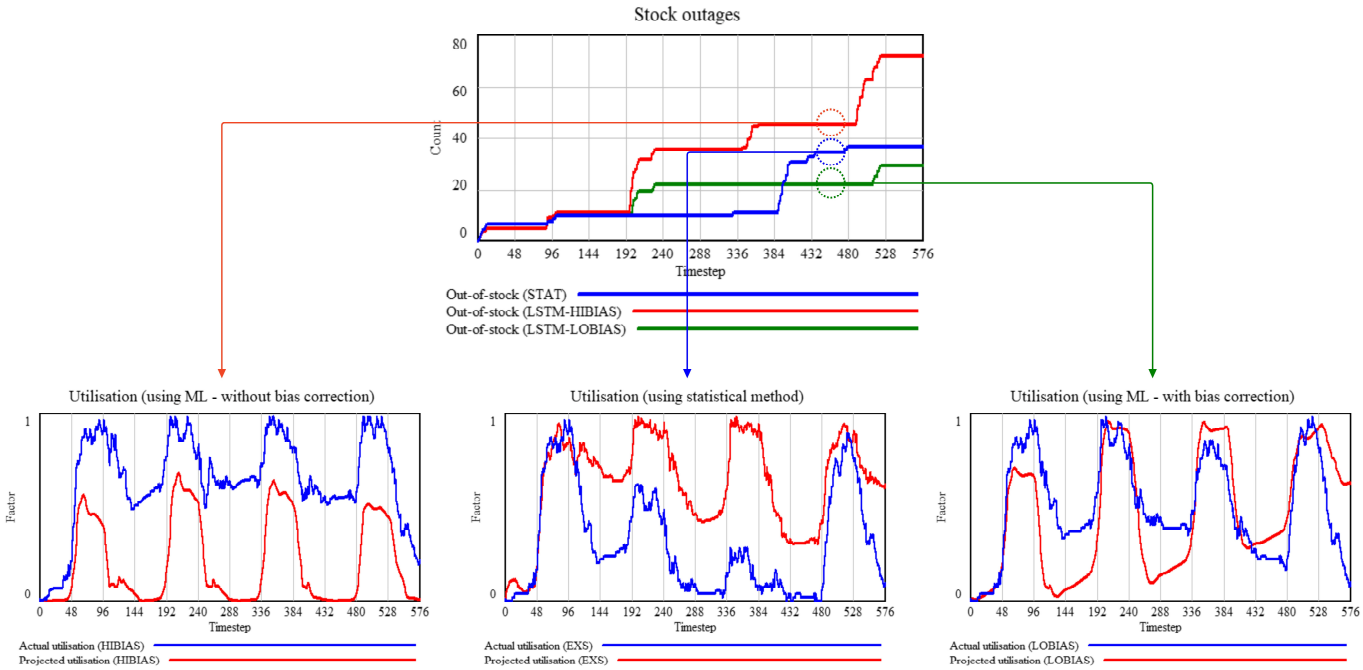


Figure 13. Bias correction analysis in the partial testing phase to enhance ML forecast accuracy and reduce systematic errors.

The projected utilisation, defined as the ratio of projected stock to maximum capacity, is consistently and significantly lower for ML than the actual utilisation, indicating a systematic bias in the predictions. Specifically, this involves either under-forecasting returns or over-forecasting rentals or both.

In this case, returns were severely under-forecasted, leading to just-in-time receipts that caused dock outages (as receipts exceeded maximum capacity). The partial test generally revealed complex temporal interdependencies between rentals and returns. These dependencies became markedly pronounced after running partial tests with zero initial stock. For instance, the station's performance is susceptible to bias and accuracy early in the simulation. Around 9 AM, returns briefly outstrip rentals, allowing a short-term build-up of inventory that reduces sensitivity to accuracy later on. This temporal pattern

appeared similarly across other stations in our demand cluster. To address the issue of bias, we adapted the LSTM model by adding a bias correction layer and incorporating a hybrid loss function that combines MSE (sensitive to large errors) and Mean Absolute Error (MAE, which measures the overall magnitude of errors). Table 1 outlines the adjustments.

Table 1. Overview of adjustments to the LSTM model for bias correction.

Procedure	Description
Bias Correction Layer	
1. Initialise the Layer	Create a trainable bias term initialised to zero.
2. Forward Pass (Call Method)	For each prediction, adjust it by adding the bias term.
3. Configuration Retrieval	Return the layer configuration.
Asymmetric Hybrid Loss Function	
1. Initialise the Loss Function	Parameters: alpha: Weight for combining MSE and MAE. lambda_bias: Penalty for bias correction. lambda_over: Penalty for over-predictions. lambda_under: Penalty for under-predictions. local_corr: Whether to compute bias correction locally (per sample) or globally (batch-wise).
2. Compute the Loss	Calculate prediction errors: $Y_{pred} = Y_{true}$. Separate errors into over-predictions (positive) and under-predictions (negative). Calculate MAE with different penalties for over- and under-predictions. Calculate MSE for all errors. Combine MSE and MAE using the weighting parameter α.
3. Bias Correction	Calculate the bias correction term, either locally (sample-wise) or globally (batch-wise). Add a penalty for bias to the loss, scaled by λ bias.
4. Return Final Loss	Final loss is the weighted sum of MSE, MAE and the bias correction penalty.
5. Configuration Retrieval	Return the loss function configuration, including all hyperparameters.

Table 2 below compares the MSE and bias across three scenarios: exponential smoothing (STAT), ML without bias correction (ML HI-B) and ML with bias correction (ML LO-B).

Table 2. Accuracy and bias metrics for the three scenarios in partial model testing.

Model	MSE Rentals	MSE Returns	Bias Rentals	Bias Returns
STAT	1.46	1.27	0	59
ML HI-B	1.05	1.03	18	−59
ML LO-B	1.05	1.02	17	29

The green line in the outage and utilisation charts in Figure 13 represents the ML LO-B scenario. Among all scenarios, this shows the closest match between projected and actual stock. While the MSE, in this case, is nearly the same as without bias correction (showing about a 28% improvement for rentals and 19% for returns), the improvement in the outage metric is 22%, compared to a 95% worse performance without the bias correction.

3.6. Evaluation—Integration and Policy Analysis

In contrast to partial testing, which emphasises the rationality of one or more subsystems (i.e., procedural rationality), whole-model testing incorporates all problem-relevant

interactions between subsystems. These inter-subsystem interactions are crucial for assessing whether the simplifying assumptions driving the design of decision rules remain valid in light of the system's emergent dynamic complexity.

A common artefact of dynamic complexity is policy resistance, which occurs when subsystem goals pull in different directions, leading to behaviours contradicting overarching system goals. Policy resistance is one example of an archetype in systems literature, defined as "common patterns of problematic behaviour" ([28], p. 111).

While there are many archetypes, policy resistance is the most relevant for assessing the impact of ML interventions on whole-system behaviour, particularly when algorithms and human judgment interact. Since ML is less likely to be influenced by existing organisational beliefs [24], it is more open to exploration, potentially placing the organisation in an unexplored decision space. However, this novelty—seen as beneficial within the limited scope of the ML model—might induce resistance elsewhere in the system. Moreover, nonlinearities and delays, typical of complex systems, mean that resistance may not manifest immediately, often producing a "better before worse" effect ([29], p. 59). Our approach in this final step of the bike-share case builds on this insight.

Based on assumptions about the future state mental model, we vary control variables that inform the judgmental component to map the fitness or performance landscape. Although constrained by structural assumptions, this landscape allows us to survey interdependencies between decision variables, providing insights into the system's "ruggedness". Ruggedness refers to the extent to which decision performance, influenced by a parameter's value, depends on other parameters. More importantly, this overview highlights leverage points in the system—policy variables that have an outsized influence on performance.

In real-life scenarios where the focal organisation knows the decision variables of routines complementing the intervention, this approach enables an evaluation of performance for both the current configuration and neighbouring parameter values. Instead of producing a single-point estimate, the organisation gains insights into the performance gradient—the direction and rate of change in performance as decision variables shift—enabling more informed decisions about what else needs to change alongside the intervention to maximise system performance.

3.6.1. Starting Inventory Policy

A natural candidate for a high-leverage control variable is the station's starting inventory for the day. Since overnight balancing actions are standard practice, setting an optimal starting inventory can significantly reduce the need for extraordinary measures, such as offering incentives or deploying the provider's fleet for additional balancing.

The performance comparison between LSTM and the statistical method showed that LSTM significantly reduces residual errors by detecting patterns from past demand and, crucially, other demand correlates. Therefore, when setting the starting inventory, LSTM's superior pattern recognition must be considered, in addition to the uncertainty introduced by randomness and unaccounted-for demand factors.

This dual requirement leads to a straightforward approach for setting the starting inventory. Forecast errors are assumed to follow a normal distribution with zero mean (indicating no systematic bias) and a standard deviation equal to the Root Mean Squared Error (RMSE). Based on a confidence interval, random errors are generated and scaled by the RMSE to reflect forecast uncertainty. Adding these errors to the forecast creates multiple demand scenarios for rentals and returns. These scenarios are then applied in a simplified simulation model (single station, without feedback loops for user incentives) to determine the most robust inventory value. Along with the confidence interval and the number of scenarios, the relative weighting of bike and dock outages is an essential parameter in evaluating performance.

Figure 14 below shows demand scenarios for the station "E 16 St and 5 Ave", generated for 24 August, with a confidence interval of 80% and 300 scenarios. Since the provider must determine the target inventory for overnight balancing before the day begins, the

forecasts are generated on a rolling basis. As much as possible, actual historical demands are used (adjusted to account for the time needed to execute physical balancing actions). However, when forecasting into the future—where actual demand data is unavailable—the previously generated forecasts are used instead. This approach enables the creation of a continuous forecast time series covering the entire day.

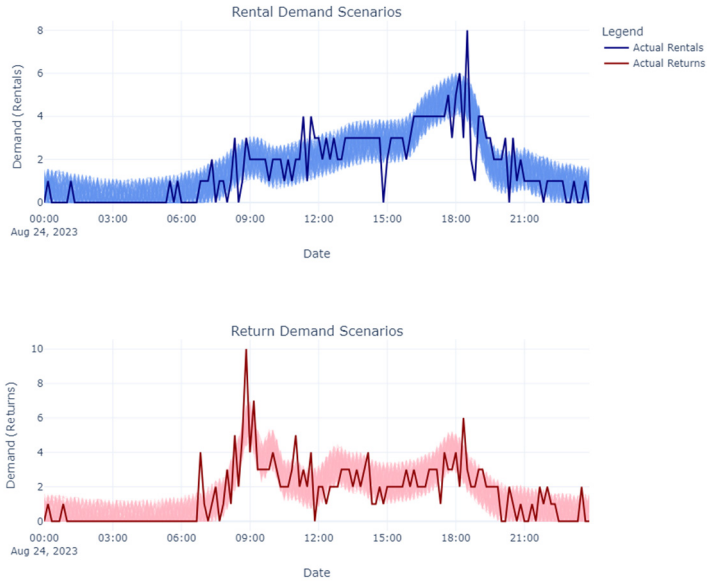


Figure 14. Generated demand variability scenarios for testing station response at "E 16 St and 5 Ave" on 24 August.

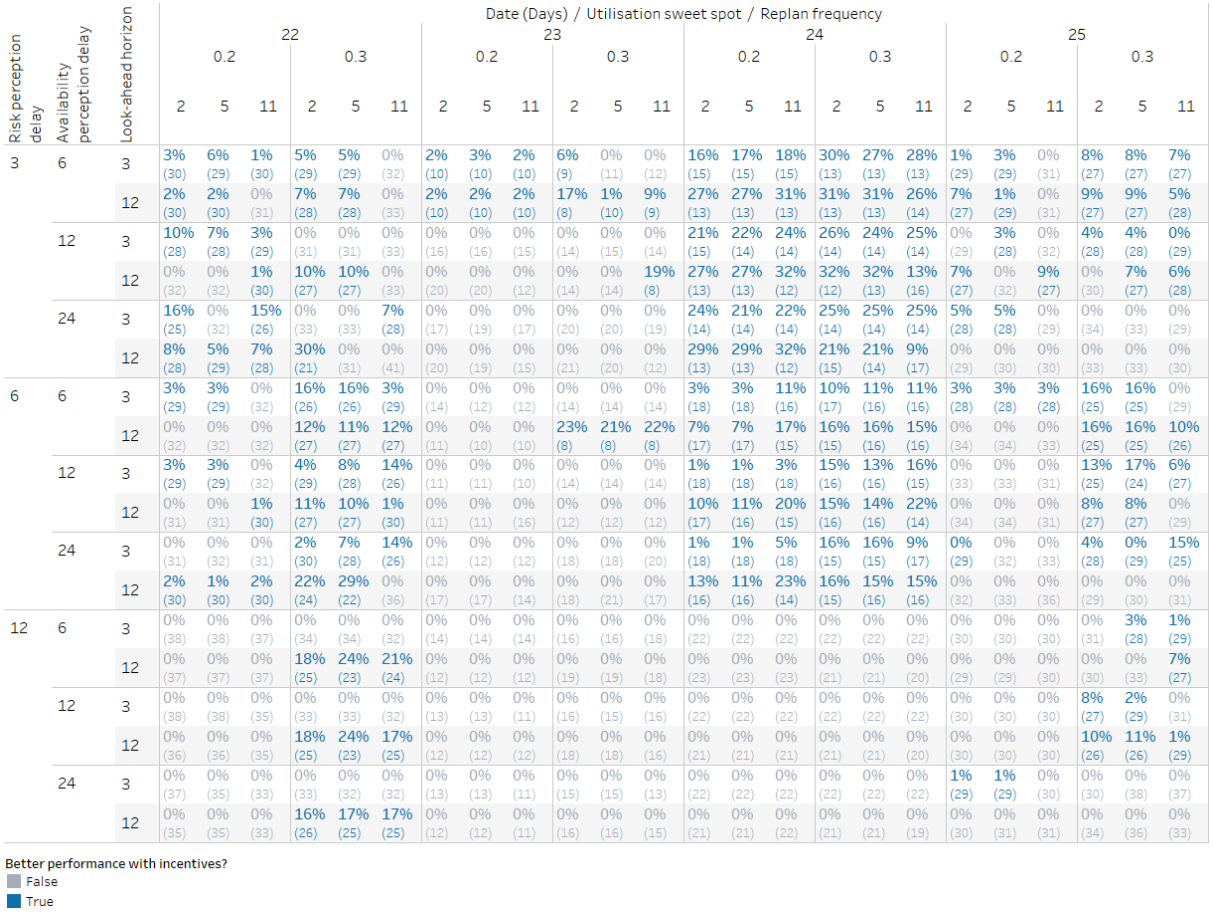
3.6.2. Simulation of Performance Across the Fitness Landscape

The dimensions of the fitness landscape—shaped by the parameters of heuristics that complement algorithmic decisions—include several decision variables beyond starting inventory. These variables include delays that influence the PeBAF and the perception of out-of-stock (OOS) risk (due to bike or dock unavailability); the aggregation logic and horizon for the PrBAF, which, along with the actual bike availability factor (BAF), affects availability perception; the refresh frequency for updating forecasts and PrBAF throughout the day; and the desired bandwidth for bike availability (where, if the PeBAF remains within the "utilisation sweet spot", inventory balancing actions are unnecessary). Each parameter set consists of specific values for these decision variables. After defining reasonable domains for the decision variables, parameter sets are generated to correspond to different policy scenarios. The simulation runs on each scenario; the results are collated and visualised in Tableau.

Given the limited incentives data (available only for August 2023), only two stations—"E 16 St and 5 Ave" and "E 16 St and Irving Pl"—qualified as candidates. These stations had incentive data and exhibited asymmetric demand patterns, as the causal inference analysis identified. We ran the policy scenarios on these stations, designated as yin and yang, respectively. The focus was on scenarios with zero starting inventory because, during the test period (22 to 25 August), the proposed target starting inventory was sufficient to prevent most stockouts. The natural buffer that builds up when returns exceed rentals further reduces the likelihood of stockouts. This buffer, however, obscured performance differences that could arise from varying parameter values. By focusing on zero-starting-inventory scenarios, we could better observe the impact of different parameters on performance, as any existing buffer would have masked these effects.

Figure 15 below presents the results of the simulation runs for parameter sets that include the decision variables mentioned earlier. The risk and availability perception delays each take three values (roughly corresponding to low, medium, and high delays); the look-ahead horizon has two values (short and long); the replan frequency is set to 2,

5 or 11 (indicating how many times the plan is refreshed throughout the day); and the utilisation sweet spot is either 20–80% (0.2) or 30–70% (0.3).



Risk perception delay	Availability perception delay	Look-ahead horizon	Date (Days) / Utilisation sweet spot / Replan frequency																							
			22						23						24						25					
			0.2			0.3			0.2			0.3			0.2			0.3			0.2			0.3		
			2	5	11	2	5	11	2	5	11	2	5	11	2	5	11	2	5	11	2	5	11	2	5	11
3	6	3	3% (30)	6% (29)	1% (30)	5% (29)	5% (29)	0% (32)	2% (10)	3% (10)	2% (10)	6% (9)	0% (11)	0% (12)	16% (15)	17% (15)	18% (15)	30% (13)	27% (13)	28% (13)	1% (29)	3% (29)	0% (31)	8% (27)	8% (27)	7% (27)
		12	2% (30)	2% (30)	0% (31)	7% (28)	7% (28)	0% (33)	2% (10)	2% (10)	2% (10)	17% (8)	1% (10)	9% (9)	27% (13)	27% (13)	31% (13)	31% (13)	31% (13)	26% (14)	7% (27)	1% (29)	0% (31)	9% (27)	9% (27)	5% (28)
	12	3	10% (28)	7% (28)	3% (29)	0% (31)	0% (31)	0% (33)	0% (16)	0% (16)	0% (15)	0% (14)	0% (15)	0% (14)	21% (15)	22% (14)	24% (14)	26% (14)	24% (14)	25% (14)	0% (29)	3% (28)	0% (32)	4% (28)	4% (28)	0% (29)
		12	0% (32)	0% (32)	1% (30)	10% (27)	10% (27)	0% (33)	0% (20)	0% (20)	0% (12)	0% (14)	0% (14)	19% (8)	27% (13)	27% (13)	32% (12)	32% (12)	32% (13)	13% (16)	7% (27)	0% (32)	9% (27)	0% (30)	7% (27)	6% (28)
	24	3	16% (25)	0% (32)	15% (26)	0% (33)	0% (33)	7% (28)	0% (17)	0% (19)	0% (17)	0% (20)	0% (20)	0% (19)	24% (14)	21% (14)	22% (14)	25% (14)	25% (14)	25% (14)	5% (28)	5% (28)	0% (29)	0% (34)	0% (33)	0% (29)
		12	8% (28)	5% (29)	7% (28)	30% (21)	0% (31)	0% (41)	0% (20)	0% (19)	0% (15)	0% (21)	0% (20)	0% (12)	29% (13)	29% (13)	32% (12)	21% (15)	21% (14)	9% (17)	0% (29)	0% (30)	0% (30)	0% (33)	0% (33)	0% (30)
6	6	3	3% (29)	3% (29)	0% (32)	16% (26)	16% (26)	3% (29)	0% (14)	0% (12)	0% (12)	0% (14)	0% (14)	0% (14)	3% (18)	3% (18)	11% (16)	10% (17)	11% (16)	11% (16)	3% (28)	3% (28)	3% (28)	16% (25)	16% (25)	0% (29)
		12	0% (32)	0% (32)	0% (32)	12% (27)	11% (27)	12% (27)	0% (11)	0% (10)	0% (10)	23% (8)	21% (8)	22% (8)	7% (17)	7% (17)	17% (15)	16% (15)	16% (16)	15% (16)	0% (34)	0% (34)	0% (33)	16% (25)	16% (25)	10% (26)
	12	3	3% (29)	3% (29)	0% (32)	4% (29)	8% (28)	14% (26)	0% (11)	0% (11)	0% (10)	0% (14)	0% (14)	0% (14)	1% (18)	1% (18)	3% (18)	15% (16)	13% (16)	16% (15)	0% (33)	0% (33)	0% (31)	13% (25)	17% (24)	6% (27)
		12	0% (31)	0% (31)	1% (30)	11% (27)	10% (27)	1% (30)	0% (11)	0% (11)	0% (16)	0% (12)	0% (12)	0% (12)	10% (17)	11% (16)	20% (15)	15% (16)	14% (16)	22% (14)	0% (34)	0% (34)	0% (31)	8% (27)	8% (27)	0% (29)
	24	3	0% (31)	0% (32)	0% (31)	2% (30)	7% (28)	14% (26)	0% (12)	0% (12)	0% (12)	0% (18)	0% (18)	0% (20)	1% (18)	1% (18)	5% (18)	16% (15)	16% (15)	9% (17)	0% (29)	0% (32)	0% (33)	4% (28)	0% (29)	15% (25)
		12	2% (30)	1% (30)	2% (30)	22% (24)	29% (22)	0% (36)	0% (17)	0% (17)	0% (14)	0% (18)	0% (21)	0% (17)	13% (16)	11% (16)	23% (14)	16% (15)	15% (16)	15% (16)	0% (32)	0% (33)	0% (36)	0% (29)	0% (30)	0% (31)
12	6	3	0% (38)	0% (38)	0% (37)	0% (34)	0% (34)	0% (32)	0% (14)	0% (14)	0% (14)	0% (16)	0% (16)	0% (18)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	0% (30)	0% (30)	0% (30)	0% (31)	3% (28)	1% (29)
		12	0% (37)	0% (37)	0% (37)	18% (25)	24% (23)	21% (24)	0% (12)	0% (12)	0% (12)	0% (19)	0% (19)	0% (18)	0% (23)	0% (23)	0% (23)	0% (21)	0% (21)	0% (20)	0% (29)	0% (29)	0% (30)	0% (30)	0% (33)	7% (27)
	12	3	0% (38)	0% (38)	0% (35)	0% (33)	0% (33)	0% (32)	0% (13)	0% (13)	0% (11)	0% (16)	0% (15)	0% (16)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	0% (30)	0% (30)	0% (30)	8% (27)	2% (29)	0% (31)
		12	0% (36)	0% (36)	0% (35)	18% (25)	24% (23)	17% (25)	0% (12)	0% (12)	0% (12)	0% (18)	0% (18)	0% (16)	0% (21)	0% (21)	0% (21)	0% (21)	0% (21)	0% (20)	0% (30)	0% (30)	0% (30)	10% (26)	11% (26)	1% (29)
	24	3	0% (37)	0% (35)	0% (33)	0% (33)	0% (32)	0% (32)	0% (13)	0% (13)	0% (11)	0% (15)	0% (15)	0% (13)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	0% (22)	1% (29)	1% (29)	0% (30)	0% (30)	0% (38)	0% (37)
		12	0% (35)	0% (35)	0% (33)	16% (26)	17% (25)	17% (25)	0% (12)	0% (12)	0% (11)	0% (16)	0% (16)	0% (15)	0% (21)	0% (21)	0% (22)	0% (21)	0% (21)	0% (19)	0% (30)	0% (31)	0% (31)	0% (34)	0% (36)	0% (33)

Better performance with incentives?
False
True

Figure 15. Performance results from simulation runs across 432 policy scenarios with decision parameters varied.

Each cell represents a policy variant, with the overview displaying 432 scenarios in total. The cell value, formatted as x% (y), shows the improvement over the scenario without crowdsourced incentives (x), along with the total number of stockouts (bikes and docks combined, y). Parameter combinations that perform better than the corresponding scenario without incentives are highlighted in blue.

3.7. Insights and Implications

A notable insight from the overview—evident in Figure 15, where the block of scenarios with the highest risk perception delay (risk delay for short) value has few highlighted cells—is the sharp decline in scenarios where crowdsourced inventory balancing through incentives outperforms the baseline (scenarios without incentives). When the risk perception delay is set to 3 and 6, 89 and 81 scenarios outperform the baseline, respectively, but this number drops significantly to 19 when the delay reaches 12.

This delay parameter is part of the anchor-and-adjust heuristic that updates risk perception. A longer delay implies a slower assimilation of the current PeBAF. The steep drop at the highest delay value suggests that the wait-and-see approach, which anchors the decision maker's risk perception to prior beliefs, results in missed intervention opportunities when availability shifts away from the desired bandwidth.

A closer inspection of scenarios with a utilisation sweet spot between 30% and 70% (risk buffer = 0.3) reveals a surprising behaviour: the number of scenarios showing improvement over the baseline slightly increases as the delay rises from 3 to 6. We analyse

the underlying stocks and flows driving this behaviour to understand this. The following graphs in Figure 16 illustrate two specific scenarios on a chosen day, comparing key stock and flow variables while holding all parameters constant except for the risk delay, set at three and six, respectively.

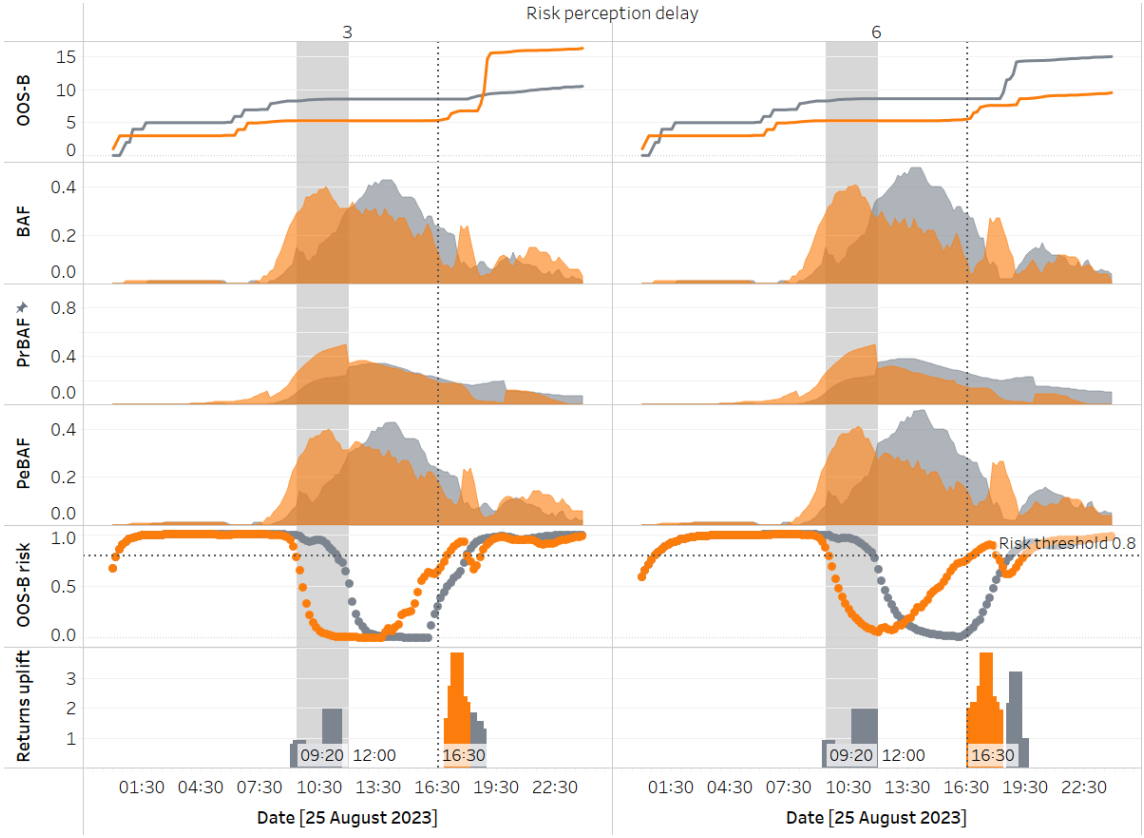


Figure 16. Analysis of key stock and flow variables influencing performance at two different risk perception delay values (orange for the yin cluster, grey for the yang cluster).

When the risk delay is set to six timesteps (or 60 min), this medium delay causes lower responsiveness to PeBAF, allowing the memory of historical risk to persist longer than with a low delay of three timesteps. Early in the day, this results in the risk of bike outages at yang remaining above the incentive threshold (indicated by the horizontal dotted line) for an extended period, prompting incentives that encourage users to return bikes to yang over yin. Later in the day, this rerouting causes yin to reach a critical stock level sooner, triggering incentives that bring in additional returns just in time to prevent most stockouts—an advantage not achieved in the low-delay scenario.

The analysis highlights how endogenous decision variables mediate the impact of exogenous temporal demand dynamics at the stations (which influence projected stock) on performance. Specifically, the inventory buffer at yin builds up faster than at yang earlier in the day. The longer delay creates a broader window for incentivisation at yang, setting off the downstream behaviours described earlier and ultimately leading to better performance under a longer delay in perceiving risk changes. While faster responsiveness to risk is generally beneficial, this example shows that uncertainty can simultaneously cause both clusters to experience high risk. In such cases, the system may defy typical assumptions about how decision variables affect performance, as recovery depends on exogenous factors.

After risk delay, the delay in assimilating real-time availability information into perception (availability delay) has the next most significant impact on performance. While this generally holds, performance varies considerably across different availability delay

values between days. For instance, the coefficient of variation in improvement over baseline (defined as the standard deviation over the mean of the improvement metric) is lowest on the 24th, at 0.85, and highest on the 23rd, at 2.92. Analysing the lowest and highest availability delays for a given set of parameters further clarifies the underlying dynamics (see Figure 17).

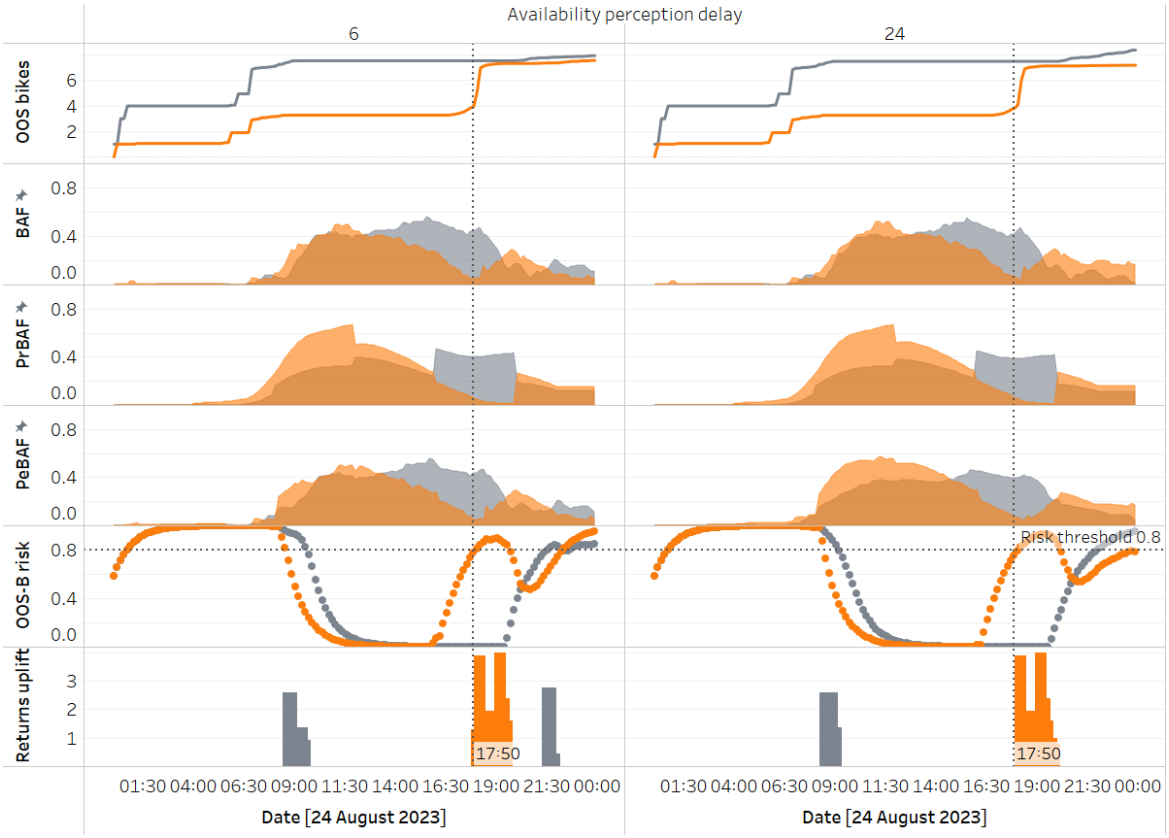


Figure 17. Analysis of key stock and flow variables influencing performance at two different availability perception delay values (orange for the yin cluster, grey for the yang cluster).

While poor forecast accuracy emphasises the need for better responsiveness to improve performance, a less obvious consequence is that it also introduces stronger interdependencies between decision variables and performance. This is illustrated on the 23rd with the decision variable aggregation horizon, which controls how many periods into the future are considered when computing PrBAF. The domain includes two values: 3 and 12 (short and long horizons).

On this day, among the scenarios that perform better than the baseline (eight in total), only one does so with a short look-ahead horizon. Due to poor forecast accuracy, averaging over a longer horizon, combined with asymmetrical demand profiles (distinct ramp-up and ramp-down patterns at the two clusters), enables initial inventory balancing via crowdsourced incentives that target the more inventory-starved cluster. As in an earlier example, this causes the source of the initial rerouting to reach a critical stock situation earlier (if at all) than in the shorter-horizon scenario. The longer horizon exacerbates this by presenting a more pessimistic view of inventory during the later ramp-down phase. In the example shown, yin is the source of the initial transfer. By intervening early enough (additional returns due to incentives starting after 17:10), yin benefits from inflows rerouted from yang, avoiding most of the stockouts it would have faced with a shorter horizon.

However, the longer look-ahead horizon is beneficial only when paired with a short availability delay (seven of the eight scenarios outperforming the baseline have a short delay). With a longer delay, PeBAF fails to assimilate inventory updates quickly enough

when balancing actions occur, leading to prolonged incentivisation at yin and causing yang's risk to exceed the threshold. Once yin's improved availability registers in PeBAF, the flow reverses direction. If the second transfer window opens early, this flipping may happen multiple times, resulting in poor stability and worse performance overall.

The increased interdependency between decision variables under poor forecast accuracy leads to greater variability in performance across scenarios, as demonstrated by the analysis of the 23rd. In contrast, on the 24th, when forecast accuracy is highest among the test days, the variability across scenarios is lowest. Since understanding these interdependencies is crucial in devising better policies, even when forecast accuracy is generally high, introducing variability by perturbing demands can provide valuable insights into their impact.

The test workbench includes an option to create demand scenarios by manipulating original demands in predefined ways. For example, we simulate scenarios where 80% of rental demands on both yin and yang stations (with asymmetrical demand patterns) until 10:00 are redistributed across the remaining periods in proportion to the demands during those periods. Simulations for these scenarios resulted in a coefficient of variance in performance improvement (considering low and medium risk delays) of 0.62 with perturbations, compared to 0.36 without (Figure 18 illustrates only the yin station for clarity). Simulations for these scenarios resulted in a coefficient of variance in performance improvement (considering low and medium risk delays) of 0.62 with perturbations, compared to 0.36 without. This variability-inducing approach is an additional tool for studying performance sensitivity to different combinations of decision variables. While test data may naturally contain high variability, this option proves helpful when variability is low or needs amplification, offering a way to explore system dynamics more thoroughly.

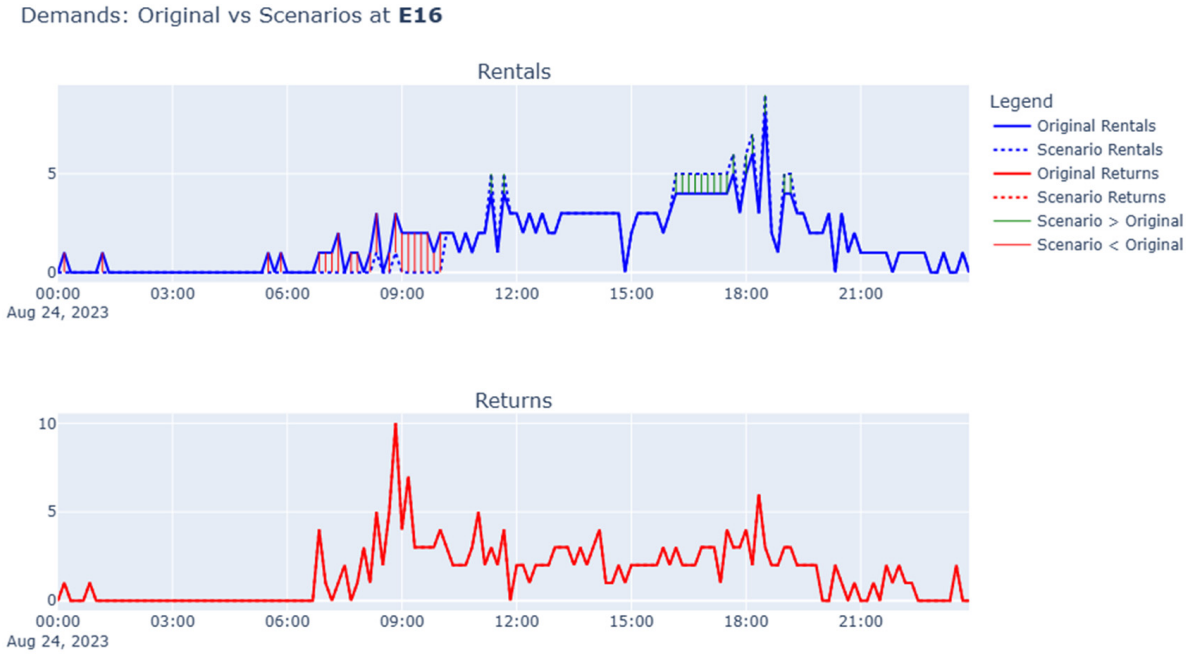


Figure 18. Impact of demand perturbation until 10 AM on station performance, with the figure illustrating results for the yin cluster.

4. Conclusions

Successive innovations in AI, particularly transformer-based models, along with increased investments in data and computing, continue to drive rapid growth in the field. These advancements create new opportunities for automation but also introduce novel risks, highlighting the need for human oversight. Recognising these risks and the potential for new tasks through human–AI collaboration has shifted the focus towards augmentation rather than pure automation in decision-making. Although frameworks for augmentation

offer guidelines—often drawing on decision theory, systems theory and empirical data—they remain only a starting point, as each organisation's approach is shaped by its unique decision-making routines and resources. To bridge the gap between theory and practice, organisations need to evaluate their specific blend of human judgment, AI and, more broadly, algorithmic decision-making. The system dynamics-based simulative modelling framework proposed in this paper addresses the "last mile" challenge of moving from a hypothesised teaming of human and AI agents to practical implementation, enabling quantification that accounts for firm-specific decision complements.

We applied the model to the inventory balancing problem in docked bike-share systems, where incentivising users to perform balancing, although eco-friendly, poses a coordination challenge. The model leverages the complementarity between stations or station clusters with asymmetrical demand patterns to optimise inventory management. Our experiments support recent research suggesting that introducing ML as a novel learning agent creates a decision-making landscape that is likely to be rugged. The flexibility of ML, unconstrained by the prior beliefs that shape human decision-making, allows it to explore a broader decision space. This flexibility creates opportunities for substantial improvement over traditional approaches but also introduces risks. For instance, our simulations of multiple policy variants for inventory balancing, combining judgmental heuristics with data science approaches (including deep learning), showed that, while ML's superior ability to anticipate future flows often leads to significant improvements, there are also scenarios where performance declines compared to using no incentives.

Furthermore, by simulating the assumed future-state policy variant and neighbouring scenarios, our approach reveals that ruggedness can result in significant variability: a variant performing well above the baseline may have nearby scenarios performing much worse. Given irreducible uncertainty, prioritising robust scenarios over elusive optimal ones is essential. Our approach enables the mapping of the performance landscape, offering the opportunity to develop robust policies through parameter fine-tuning and structural adjustments that deliver synergistic human–AI teaming.

Future work could extend the analysis to an entire network of stations, leveraging the scalability of our PySD-based simulation workbench to run parallel simulations for multiple demand clusters. Incorporating structural changes to the model, such as the ability to simulate carrier-based rebalancing alongside crowdsourced strategies, would add realism and support a more comprehensive analysis of balancing measures. Additionally, validating the assumptions underpinning perceived availability and risk—as well as other heuristics that synthesise algorithmic predictions and human judgment—would be essential when implementing the framework in real-world settings. These adaptations would need to align with firm-specific decision-making routines to ensure applicability across different operational environments. Finally, while this study focused on bike-sharing, the framework could be adapted to other complex systems requiring a blend of human and machine-led decision-making, such as supply chain planning and resource allocation problems.

Author Contributions: Conceptualization, G.S. (Ganesh Sankaran) and M.A.P.; Methodology, G.S. (Ganesh Sankaran); Software, G.S. (Ganesh Sankaran); Validation, G.S. (Ganesh Sankaran) and M.A.P.; Investigation, G.S. (Ganesh Sankaran); Writing—original draft, G.S. (Ganesh Sankaran); Writing—review & editing, M.A.P. and G.S. (Guido Siestrup); Visualization, G.S. (Ganesh Sankaran); Supervision, M.A.P., M.K. and G.S. (Guido Siestrup); Project administration, M.A.P. and G.S. (Guido Siestrup); Funding acquisition, G.S. (Guido Siestrup). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The ML code, system dynamics simulation files and data are available on GitHub under: bss-model-E3BF (https://anonymous.4open.science/r/bss-model-E3BF (accessed on 10 October 2024)).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Bike-share forecast model characteristics.

Aspect	Description
Model Architecture	- Sequential model - Input layer - Two LSTM layers (64 units each, ReLU activation) - Custom Bias Correction layer (corrects bias in predictions) - Dense output layer
Features Used	- Rentals- Returns - Returns as correlate for rentals forecast - Rentals as correlate for returns forecast - Time signals (sin/cos of time of day and week)
Optional Features	- Weather data (actual or forecast) - Lagged correlates - Contemporaneous correlates
Model Parameters	- Sequence length: 288 (2 days $\times$ 24 h $\times$ 6 records per hour) - Forecast horizon: 12 (2 h $\times$ 6 records per hour)- Batch size: 24 - LSTM units: 64 per layer - L1 regularisation: 1×10^{-4}- L2 regularisation: 1×10^{-5} - Learning rate: 0.002 (with ReduceLROnPlateau)
Loss Function	- Asymmetric Hybrid Loss with parameters: α, λ_bias, λ_over and λ_under (custom loss function designed to penalise different types of errors differently)
Optimisation	- Adam optimiser- ReduceLROnPlateau (factor: 0.8, patience: 5 epochs, min_lr: 0.0003)
Data Preprocessing	- MinMax scaling (feature range: -1 to 1) - Sequence generation using Keras utilities - Optional differencing (24 $\times$ 6 periods)
Training Process	- Early stopping (monitoring validation loss, patience: 10 epochs) - ReduceLROnPlateau (learning rate reduction on plateau)- Maximum 50 epochs
Evaluation Metrics	- MSE, MAE
Implementation	- TensorFlow/Keras
Data Granularity	- 10 min intervals (applies to both input data and output predictions)
Forecasting Approach	- Separate models for rentals and returns
Notable Features	- Custom Bias Correction layer (for improved prediction accuracy) - Asymmetric Hybrid Loss function (penalises under- and over-estimations differently) - Flexibility to include/exclude various features - Sequence-to-sequence option
Reproducibility	- TensorFlow version: 2.17.0 - Keras version: 3.4.1 - Python version: 3.10.12 - GPU: Tesla T4 with CUDA 12.2 - Fixed random seed (511) for TensorFlow and NumPy operations

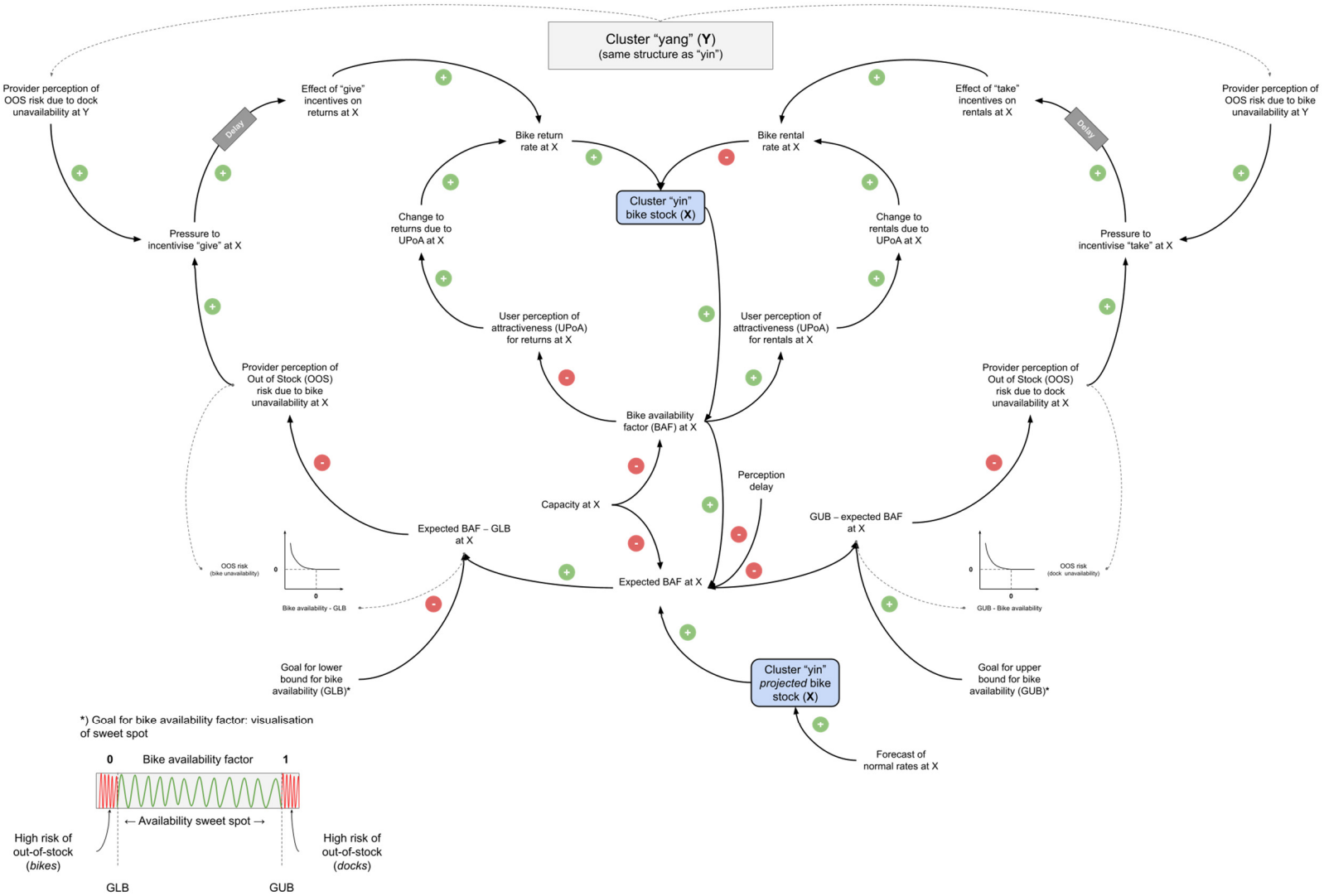


Figure A1. CLD of the bike-share two-stock model.

References

1. Raschka, S. *Build a Large Language Model from Scratch*; Manning Publications: Shelter Island, NY, USA, 2024; 400p.
2. Malone, T.W. How Human-Computer "Superminds" Are Redefining the Future of Work. *MIT Sloan Manag. Rev.* **2018**, *59*, 34–41.
3. Agrawal, A.; Gans, J.S.; Goldfarb, A. What to Expect from Artificial Intelligence. *MIT Sloan Manag. Rev.* **2017**, *58*, 23. Available online: https://sloanreview-mit-edu.plymouth.idm.oclc.org/article/what-to-expect-from-artificial-intelligence/ (accessed on 14 September 2021).
4. Saenz, M.J.; Revilla, E.; Simón, C. Designing AI Systems With Human-Machine Teams. *MIT Sloan Manag. Rev.* **2020**, *61*, 1–5. Available online: https://sloanreview.mit.edu/article/designing-ai-systems-with-human-machine-teams/ (accessed on 8 September 2021).
5. Raisch, S.; Krakowski, S. Artificial Intelligence and Management: The Automation–Augmentation Paradox. *AMR* **2021**, *46*, 192–210. [CrossRef]
6. Brynjolfsson, E.; Mitchell, T. What can machine learning do? Workforce implications. *Science* **2017**, *358*, 1530–1534. [CrossRef]
7. Autor, D. *Polanyi's Paradox and the Shape of Employment Growth*; Report No.: 20485; National Bureau of Economic Research: Cambridge, MA, USA, 2014. Available online: https://www.nber.org/papers/w20485 (accessed on 8 September 2021).
8. Agrawal, A.; Gans, J.; Goldfarb, A. *Power and Prediction: The Disruptive Economics of Artificial Intelligence*; Harvard Business Review Press: Boston, MA, USA, 2022; 288p.
9. Brynjolfsson, E. The Turing Trap: The Promise & Peril of Human-Like Artificial Intelligence. *Daedalus* **2022**, *151*, 272–287. [CrossRef]
10. Acemoglu, D.; Johnson, S. *Power and Progress: Our Thousand-Year Struggle over Technology and Prosperity*, 1st ed.; Public Affairs: New York, NY, USA, 2023; 560p.
11. Kambhampati, S. Polanyi's Revenge and AI's New Romance with Tacit Knowledge. *Commun. ACM* **2021**, *64*, 31–32. [CrossRef]
12. Lebovitz, S.; Levina, N.; Lifshitz-Assaf, H. Is AI Ground Truth Really "True?" The Dangers of Training and Evaluating AI Tools Based on Experts' Know-What. *Manag. Inf. Syst. Q.* **2021**, *45*, 1501–1526. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3839601 (accessed on 28 October 2022). [CrossRef]
13. Raghu, M.; Blumer, K.; Corrado, G.; Kleinberg, J.; Obermeyer, Z.; Mullainathan, S. The Algorithmic Automation Problem: Prediction, Triage, and Human Effort. *arXiv* **2019**, arXiv:1903.12220.
14. Ross, J. Don't Confuse Digital with Digitization. *MIT Sloan Manag. Rev.* 2019. Available online: https://sloanreview.mit.edu/article/dont-confuse-digital-with-digitization/ (accessed on 7 November 2022).
15. Moser, C.; Hond, F.; den Lindebaum, D. What Humans Lose When We Let AI Decide. *MIT Sloan Manag. Rev.* **2022**, *63*, 12–14.
16. Morgan, G. *Images of Organization*; Updated edition; SAGE Publications, Inc.: Thousand Oaks, CA, USA, 2006; 520p.
17. Gigerenzer, G. *How to Stay Smart in a Smart World: Why Human Intelligence Still Beats Algorithms*; Penguin: Portland, Oregon, 2022; 307p.
18. Chiang, T. ChatGPT Is a Blurry JPEG of the Web. *The New Yorker*, 2023. Available online: https://www.newyorker.com/tech/annals-of-technology/chatgpt-is-a-blurry-jpeg-of-the-web (accessed on 24 November 2023).
19. Babic, B.; Cohen, I.G.; Evgeniou, T.; Gerke, S. When Machine Learning Goes Off the Rails. *Harv. Bus. Rev.* **2021**, 132–138. Available online: https://hbr.org/2021/01/when-machine-learning-goes-off-the-rails (accessed on 25 May 2022).
20. Smith, B.C. *The Promise of Artificial Intelligence: Reckoning and Judgment*; Illustrated Edition; The MIT Press: Cambridge, MA, USA, 2019; 184p.
21. Kitchin, R. Big Data, New Epistemologies and Paradigm Shifts. *Big Data Soc.* **2014**, *1*, 2053951714528481. [CrossRef]
22. Domingos, P. A Few Useful Things to Know About Machine Learning. *Commun. ACM* **2012**, *55*, 78–87. [CrossRef]
23. Levinthal, D.A. Adaptation on Rugged Landscapes. *Manag. Sci.* **1997**, *43*, 934–950. [CrossRef]
24. Sturm, T.; Gerlach, J.P.; Pumplun, L.; Mesbah, N.; Peters, F.; Tauchert, C.; Nan, N.; Buxmann, P. Coordinating Human and Machine Learning for Effective Organizational Learning. *MIS Q.* **2021**, *45*, 1581–1602. [CrossRef]
25. Dell'Acqua, F.; McFowland, E., III; Mollick, E.R.; Lifshitz-Assaf, H.; Kellogg, K.; Rajendran, S.; Lakhani, K.R. *Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality*. Harvard Business School Technology & Operations Mgt. Unit Working Paper. 2023. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4573321 (accessed on 13 November 2023).
26. Oliveira, G.N.; Sotomayor, J.L.; Torchelsen, R.P.; Silva, C.T.; Comba, J.L.D. Visual analysis of bike-sharing systems. *Comput. Graph.* **2016**, *60*, 119–129. [CrossRef]
27. Chung, H.; Freund, D.; Shmoys, D.B. Bike Angels: An Analysis of Citi Bike's Incentive Program. In Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS' 18), New York, NY, USA, 20–22 June 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–9. [CrossRef]
28. Meadows, D.H. *Thinking in Systems: International Bestseller*; Wright, D., Ed.; Chelsea Green Publishing: Hartford, VT, USA, 2008; 240p.
29. Senge, P.M. *The Fifth Discipline: The Art & Practice of The Learning Organization*; Revised & Updated edition; Doubleday: New York, NY, USA, 2006; 445p.

30. Forrester, J.W. Industrial Dynamics. *Harv. Bus. Rev.* **1958**, *36*, 37–66.
31. Pruyt, E. *Small System Dynamics Models for Big Issues: Triple Jump Towards Real-World Complexity*; TU Delft Library: Delft, The Netherlands, 2013; 324p.
32. Cassidy, R.; Singh, N.S.; Schiratti, P.-R.; Semwanga, A.; Binyaruka, P.; Sachingongu, N.; Chama-Chiliba, C.M.; Chalabi, Z.; Borghi, J.; Blanchet, K. Mathematical modelling for health systems research: A systematic review of system dynamics and agent-based models. *BMC Health Serv. Res.* **2019**, *19*, 845. [CrossRef]
33. Morecroft, J.D. System dynamics: Portraying bounded rationality. *Omega* **1983**, *11*, 131–142. [CrossRef]
34. Lyneis, J.M. System dynamics for market forecasting and structural analysis. *Syst. Dyn. Rev.* **2000**, *16*, 3–25. [CrossRef]
35. Vlachos, D.; Georgiadis, P.; Iakovou, E. A system dynamics model for dynamic capacity planning of remanufacturing in closed-loop supply chains. *Comput. Oper. Res.* **2007**, *34*, 367–394. [CrossRef]
36. Houghton, J.; Siegel, M. Advanced data analytics for system dynamics models using PySD. In Proceedings of the 33rd International Conference of the System Dynamics Society, Cambridge, MA, USA, 19–23 July 2015; System Dynamics Society: Cambridge, UK, 2015.
37. Chen, Y.T.; Tu, Y.M.; Jeng, B. A Machine Learning Approach to Policy Optimization in System Dynamics Models. *Syst. Res. Behav. Sci.* **2011**, *28*, 369–390. [CrossRef]
38. Edali, M. Pattern-oriented analysis of system dynamics models via random forests. *Syst. Dyn. Rev.* **2022**, *38*, 135–166. [CrossRef]
39. Simon, H.A. *The Sciences of the Artificial*, 3rd ed.; The MIT Press: Cambridge, MA, USA, 1996; 248p.
40. Sankaran, G.; Palomino, M.A.; Knahl, M.; Siestrup, G. A modeling approach for measuring the performance of a human-AI collaborative process. *Appl Sci.* **2022**, *12*, 11642. [CrossRef]
41. Caggiani, L.; Ottomanelli, M. A Dynamic Simulation based Model for Optimal Fleet Repositioning in Bike-sharing Systems. *Procedia-Soc. Behav. Sci.* **2013**, *87*, 203–210. [CrossRef]
42. Lowalekar, M.; Varakantham, P.; Ghosh, S.; Jena, S.; Jaillet, P. Online Repositioning in Bike Sharing Systems. In Proceedings of the International Conference on Automated Planning and Scheduling, Pittsburgh, PA, USA, 18–23 June 2017; Volume 27, pp. 200–208. Available online: https://ojs.aaai.org/index.php/ICAPS/article/view/13824 (accessed on 6 November 2024).
43. Legros, B. Dynamic repositioning strategy in a bike-sharing system; how to prioritise and how to rebalance a bike station. *Eur. J. Oper. Res.* **2019**, *272*, 740–753. [CrossRef]
44. Ghosh, S.; Trick, M.; Varakantham, P. Robust Repositioning to Counter Unpredictable Demand in Bike Sharing Systems. In Proceedings of the 25th International Joint Conference on Artificial Intelligence IJCAI 2016, New York, NY, USA, 9–15 July 2016; pp. 3096–3102. Available online: https://ink.library.smu.edu.sg/sis_research/3456 (accessed on 6 November 2024).
45. Sterman, J.D. *Business Dynamics*; International edition; McGraw-Hill Education: Boston, MA, USA, 2000; 993p.
46. Will, M.; Bertrand, J.; Fransoo, J.C. Operations management research methodologies using quantitative modeling. *Int. J. Oper. Prod. Manag.* **2002**, *22*, 241–264. [CrossRef]
47. Morecroft, J.D.W. *Strategic Modelling and Business Dynamics: A Feedback Systems Approach*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2015; 504p.
48. Mitroff, I.I.; Betz, F.; Pondy, L.R.; Sagasti, F. On Managing Science in the Systems Age: Two Schemas for the Study of Science as a Whole Systems Phenomenon. *Interfaces* **1974**, *4*, 46–58. Available online: https://www.jstor.org/stable/25059093 (accessed on 22 October 2021). [CrossRef]
49. Shen, Y.; Zhang, X.; Zhao, J. Understanding the usage of dockless bike sharing in Singapore. *Int. J. Sustain. Transp.* **2018**, *12*, 686–700. [CrossRef]
50. Shaheen, S.A.; Guzman, S.; Zhang, H. Bikesharing in Europe, the Americas, and Asia: Past, Present, and Future. *Transp. Res. Rec.* **2010**, *2143*, 159–167. [CrossRef]
51. Singla, A.; Santoni, M.; Bartók, G.; Mukerji, P.; Meenen, M.; Krause, A. Incentivizing Users for Balancing Bike Sharing Systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January; 2015; Volume 29. Available online: https://ojs.aaai.org/index.php/AAAI/article/view/9251 (accessed on 11 August 2023).
52. El Sibai, R.; Challita, K.; Bou Abdo, J.; Demerjian, J. A New User-Based Incentive Strategy for Improving Bike Sharing Systems' Performance. *Sustainability* **2021**, *13*, 2780. Available online: https://www.mdpi.com/2071-1050/13/5/2780 (accessed on 6 November 2024). [CrossRef]
53. Makridakis, S.G.; Wheelwright, S.C.; Hyndman, R.J. *Forecasting: Methods and Applications*, 3rd ed.; Wiley: New York, NY, USA, 1998; 923p.
54. Sankaran, G.; Sasso, F.; Kepczynski, R.; Chiaraviglio, A. *Improving Forecasts with Integrated Business Planning: From Short-Term to Long-Term Demand Planning Enabled by SAP IBP*; Management for Professionals; Springer International Publishing: Cham, Switzerland, 2019. Available online: http://link.springer.com/10.1007/978-3-030-05381-9 (accessed on 27 August 2024).
55. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd ed.; O'Reilly Media: Sebastopol, CA, USA, 2019; 856p.
56. Chollet, F. *Deep Learning with Python*, 2nd ed.; Manning: Shelter Island, NY, USA, 2021; 504p.
57. Brodersen, K.H.; Gallusser, F.; Koehler, J.; Remy, N.; Scott, S.L. Inferring causal impact using Bayesian structural time-series models. *Ann. Appl. Stat.* **2015**, *9*, 247–274. [CrossRef]

58. Morecroft, J.D.W. Rationality in the Analysis of Behavioral Simulation Models. *Manag. Sci.* **1985**, *31*, 900–916. [CrossRef]
59. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE* **2018**, *13*, e0194889. [CrossRef] [PubMed]