Decision Support Systems 147 (2021) 113584

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

A prescriptive analytics framework for efficient E-commerce order delivery

Shanthan Kandula [a,*], Srikumar Krishnamoorthy [a], Debjit Roy [a,b]

[a] Indian Institute of Management Ahmedabad, Gujarat 380015, India
[b] Rotterdam School of Management, Erasmus University, Rotterdam 3062, PA, the Netherlands

ARTICLE INFO

Keywords:
Analytics
Data-driven delivery
Machine learning
Vehicle routing
E-commerce

ABSTRACT

Achieving timely last-mile order delivery is often the most challenging part of an e-commerce order fulfillment. Effective management of last-mile operations can result in significant cost savings and lead to increased customer satisfaction. Currently, due to the lack of customer availability information, the schedules followed by delivery agents are optimized for the shortest tour distance. Therefore, orders are not delivered in customer-preferred time periods resulting in missed deliveries. Missed deliveries are undesirable since they incur additional costs. In this paper, we propose a decision support framework that is intended to improve delivery success rates while reducing delivery costs. Our framework generates delivery schedules by predicting the appropriate delivery time periods for order delivery. In particular, the proposed framework works in two stages. In the first stage, order delivery success for every order throughout the delivery shift is predicted using machine learning models. The predictions are used as an input for the optimization scheme, which generates delivery schedules in the second stage. The proposed framework is evaluated on two real-world datasets collected from a large e-commerce platform. The results indicate the effectiveness of the decision support framework in enabling savings of up to 10.2% in delivery costs when compared to the current industry practice.

1. Introduction

E-commerce is contributing significantly to retail sales. In 2019, online retailing accounted for 14.1% of the retail sales, and this figure is estimated to reach 22% by 2023 [1]. Except for digital goods, all other products sold on e-commerce platforms require physical delivery. The last-mile delivery of goods, in the context of business-to-customer retailing, includes several challenges [2]. In fact, physical delivery is the most polluting, least efficient part of the logistics [3]. One of the problems with the physical delivery of goods is the presence of failed deliveries. Since failed order deliveries are reattempted, they incur additional costs in package handling, fuel, and emissions [3]. On the other hand, successful deliveries increase customer satisfaction and decrease the chance of product returns [4]. Therefore, it is highly desirable for logistics operators and platform owners to increase successful deliveries to make the delivery process efficient, and increase customer satisfaction.

The primary reason for failed deliveries is the absence of customers when deliveries are attempted [5]. Several last-mile delivery models aim to reduce missed deliveries. One possible solution is to deliver goods to pick-up points and instruct customers to collect them. Though this solution offers the potential to reduce delivery failures, it also inconveniences the customers because they have to physically visit the pick-up points [6]. It is found that the convenience offered by home delivery is a significant driver of e-commerce purchases [7]; therefore, this solution can negatively impact sales. The other solution is to deliver goods on pre-agreed time windows. Although this solution holds promise in increasing successful deliveries, time-windows are generally not offered for ordinary deliveries (goods other than grocery, furniture, heavy appliances) [8]. One reason is the associated difficulty in knowing the exact delivery date a priori. Deliveries often lead or lag the estimated time of arrival (ETA) [9], making it challenging to offer time slots. In the current e-commerce setting where home-delivery is the preferred delivery option [10,11] and offering delivery time-windows in advance is difficult, knowing the appropriate time periods for attended home delivery would be helpful to minimize missed deliveries and improve customer satisfaction.

Predictive models can help platforms learn appropriate delivery time-windows. Learning appropriate time windows allows platforms to plan delivery schedules that result in successful deliveries. Also, identifying orders that are likely to be missed allows companies to take proactive remedial measures. For instance, customers can be contacted

* Corresponding author.
E-mail addresses: phd18shanthank@iima.ac.in (S. Kandula), srikumark@iima.ac.in (S. Krishnamoorthy), debjit@iima.ac.in (D. Roy).

https://doi.org/10.1016/j.dss.2021.113584
Received 2 November 2020; Received in revised form 18 March 2021; Accepted 27 April 2021
Available online 30 April 2021

before attempting the delivery to confirm their willingness to receive orders. Overall, predictive models can be leveraged in schedule planning to decrease delivery failures, reduce operational costs, and increase customer satisfaction.

So far, a few studies [12–15] proposed data-driven solutions for increasing delivery successes; however, the approaches suffer from drawbacks. At the prediction task level, they employ sensitive customer-specific information such as half-hourly electricity consumption data, historical GPS coordinates, and smart home device data. These data sources are not readily available, and customers do not readily share them [16–18]. Also, there are legal issues concerning the security and privacy of customers. In fact, two of the studies [12,14] acknowledge the legal issues, and one study [14] acknowledges the non-availability of data sources and calls for research on using other kinds of data. Our work explores less sensitive, readily available data sources for building predictive models. We use alternate data sources such as aggregated location data and historical order delivery data that are widely available and propose novel predictors for building predictive models. At the schedule generation level, prior approaches integrate predictions using methods that take exponential time for computation. Though these methods are useful for smaller instances, e-commerce delivery hubs deliver hundreds of orders in a day making exponential time methods inadaptable. Our work builds on the available construction heuristics, which takes polynomial time for computation. Though heuristics usage reduces solution quality, the schedules can be generated in minutes against hours (or days).

Our approach is based on the fact that the success of a delivery attempt depends on both the order characteristics [3] and the location characteristics of a service region [19]. For instance, deliveries directed to office and business locations are more likely to succeed in the afternoon hours than deliveries directed to residential areas. Order features such as delivery-speed, value, and category can suggest the customers' willingness to wait. For example, customers choosing same-day or one-day delivery options are less willing to wait [9]; hence they may make arrangements for order reception resulting in high delivery success. Similarly, several order-related features and location-related features can become helpful in predicting delivery success.

In this paper, our first aim is to demonstrate that a service region's location features and order features can be used to predict the success or failure of a delivery attempt. To this end, we train several machine learning models on order delivery data supplied by our industry collaborator and location data extracted from Here technologies' location platform [20] and compare their predictive performance. Our second aim is to design a decision support framework that leverages delivery success predictive models and generates delivery schedules to service customers in the most appropriate time intervals. Accordingly, we propose a two-step decision support framework. In the first step, the framework uses machine learning models to generate order success profiles. Order success profiles map the probability of delivery success to the time of delivery. In the second step, for each order, the framework infers time windows appropriate for the delivery and uses the Vehicle Routing Problem with Time Windows (VRPTW) optimization scheme to generate the delivery schedules.

Our framework is developed in collaboration with one of the largest e-commerce platforms in India. Currently, the delivery agents deliver orders in the sequence that results in the shortest tour distance. In contrast, our framework generates schedules identifying the most appropriate time for order deliveries, thereby increasing the chance of delivery success. We establish the validity and generality of the approach by applying it in two diverse delivery hubs and comparing the operational costs and the required delivery attempts with that of the current process in practice. Specifically, we evaluate our approach against the baseline policy, where delivery schedules are optimized to produce the shortest tour through simulation experiments. Compared to the baseline policy, our approach reduces the delivery attempts and operational costs for both the hubs. The paper's contributions are as follows:

- It demonstrates the usage of diverse data sources such as e-commerce order delivery data and location data for building delivery success prediction models.
- It is the first paper to propose order success profiles and to define a procedure to generate them.
- It presents a practical decision support framework that generates appropriate time windows for order deliveries and uses them to develop schedules.

The remainder of this paper is organized as follows. In Section 2, we review the related literature. Section 3 describes the proposed decision support framework. The case study, data utilized, and the experimental settings are described in Section 4. The results of both predictive modeling and schedule generation are discussed in Section 5. In Section 6, we discuss our findings and provide concluding remarks.

2. Literature review

2.1. Traditional order fulfillment methods

Reception boxes, controlled access systems, collection & delivery points (CDPs), attended home delivery (AHD) with time slot management are traditionally proposed in the literature to reduce missed deliveries [21]. Reception boxes are similar to letterboxes installed at customers' houses to accommodate order deliveries [2]. Since deliveries can happen irrespective of customers' presence, this solution reduces failed deliveries. However, due to the challenges it poses to business models and the associated low practical feasibility, the adoption of reception boxes is small-scale [10].

Controlled access systems such as smart locks are keyless lock systems that allow delivery agents to access private delivery locations such as car trunks and houses via mobile applications and digital keys. Similar to reception boxes, the delivery is independent of customers' presence resulting in increased successful deliveries. Though smart locks' technical feasibility is demonstrated in several parts of the world, due to the concerns of security, customer adoption is little [10].

CDPs are third-party locations that accommodate deliveries and allow customers to collect and return orders placed online. These can be a collection of lockers or the existing shops, post offices, and pharmacies. Like reception boxes and smart locks, CDPs make the delivery independent of customer availability and increase successful deliveries. Though CDPs reduce failed deliveries, they increase customer mileage [22], resulting in inconvenience to customers. In contrast to CDPs and other solutions discussed above, our approach neither requires additional infrastructure nor forces the customers to travel to a third-party location. Our framework preserves the convenience offered by the home delivery model by finding appropriate times for attended deliveries instead of directing them to alternative locations.

In AHD with time slot management, customers choose one of the available time slots during which deliveries are attempted. Since customers become aware of deliveries in advance, the probability of delivery failure decreases. Though this solution reduces the number of failed deliveries, the choice of time slots may result in uneven distribution of drop-offs, increasing route complexity, and delivery costs [23]. Therefore, to gain operational efficiency, time slot management techniques are introduced. Four management techniques are proposed in the literature: differentiated slotting, differentiated pricing, dynamic slotting, and dynamic pricing [24]. In the time slot management schemes, once the customers select time slots for a day, a VRPTW problem is solved to generate delivery routes or schedules. VRPTW problem is extensively studied; readers can refer Toth & Vigo [25] for an in-depth understanding.

In contrast to the time slot management schemes, in our approach, the time slots are not selected by the customers but are derived from the

order success profiles generated by machine learning models. The order success profiles we propose help to identify appropriate time periods for deliveries. Since time slots are not offered for ordinary deliveries (goods other than groceries, furniture & heavy appliances) [8], our approach can serve as an alternative to infer time slots and to plan in accordance. We also include a simple, scalable heuristic to solve the VRPTW problem that leverages the structure introduced by the inferred time slots.

2.2. Data-driven fulfillment methods

Our work is closely related to the recent literature on data-driven delivery fulfillment methods. Pan, Giannikas, Han, Grover-Silva, & Qiao [12] were the earliest to propose an approach that employs data-driven models for increasing successful deliveries. They use electricity consumption data along with demographic information to determine customer absence probability. The hour having the least probability of customer absence is treated as a time slot. Finally, they use the VRPTW framework to generate delivery schedules. Florio, Feillet, & Hartl [13] assume that customer availability throughout the delivery period is already known and formulate a delivery problem that considers multiple revisits to a customer to increase delivery successes. Their computational approach takes 5 hours to solve an instance size of 100 customers. Praet & Martens [14] employ customer GPS information to predict customer location. Mangiaracina, Perego, Seghezzi, & Tumino [15] use smart home device data like Wi-Fi connections, Bluetooth connections to predict the customer presence probability. The probabilities are then used in a VRP framework to generate the delivery schedules.

Our work differs from the prior solutions at two levels: one at the prediction stage and the other at the schedule generation stage. At the prediction level, out of the discussed four studies, only three studies [12,14,15] build a prediction model. All three studies utilize sensitive, customer-specific information: electricity consumption, smart home appliance data, and GPS coordinate data. These data sources are context-specific and not widely available. Obtaining customer GPS location data throughout the day is challenging. A recent study revealed that 75–80% of customers feel stressed to share their GPS locations, and overall, only 21% of them share it [16]. The authors [14] also observe that the number of active users sharing GPS data are small. Similarly, profiling customers through smart home devices may not be preferable due to legal issues surrounding privacy. For instance, 63% of customers already feel smart home devices are “creepy” [17]. Gathering electricity consumption information is even more difficult because of the requirement of smart meters. Frequent monitoring of electricity load requires smart meters [26], but the global penetration of smart meters is only 14% as of November 2019 [27]. Furthermore, the studies themselves [12,14] acknowledge the legal issues that arise from using sensitive customer information. One study [14] acknowledges that sensitive customer information such as customer GPS information is not omnipresent and calls for research on using other kinds of less sensitive data. Our work is an answer to that call. We use order delivery data and location data aggregated at the delivery location level to build prediction models. *E*-commerce platforms are currently capturing order delivery data, and location data aggregated at the delivery area level can be easily sourced from any third-party location data provider. Besides, only one study [14] reports the performance of models, and our models are more accurate than their models.

At the schedule generation level, out of the discussed four studies, only three studies generate schedules. The formulation by Florio, Feillet, & Hartl [13] is exponential in time and takes 5 hours to generate schedules for 100 customers. The formulations in the other two studies [12,15] are MILP (Mixed-integer linear programming) based, which are known to take exponential computational time. Our approach takes polynomial time and hence is scalable in the e-commerce context where a few hundreds of orders are delivered by a single hub daily. Table 1 compares previous approaches with ours.

2.3. Machine learning-based decision support frameworks

Our work is also related to the research on decision support frameworks driven by predictive machine learning. Machine learning-oriented methods are applied in a variety of settings. For instance, diverse applications such as container stacking operations [28], employee recruitment [29], bankruptcy prediction [30], R&D budget allocation [31], aviation incident risk prediction [32] utilize predictive models for decision support. In contrast to these approaches, we propose a framework combining predictive models and VRPTW optimization to generate data-driven delivery schedules. Our work builds upon the existing literature and develops an innovative decision support framework, which is an addition to the research on decision support systems.

In our approach, we use diverse data sets for building predictive models. The usage of multiple data sources is extensively studied in the information fusion literature [33]. In general, the datasets are fused at the signal level or feature level, or decision level while building machine learning models [34]. Our framework fuses data at the feature level. Specifically, we concatenate features from both the datasets and utilize them as inputs to the models we build. Feature level fusion provides complementary information and improves the accuracy of predictive models [35]. Numerous studies employ feature-level fusion. In the recent past, Picasso, Merello, Ma, Oneto, & Cambria [36] concatenate financial indicators and sentiment embeddings to predict market trends. Huang, Bergman, & Gopal [37] combine demographic features with product sales data in their analytics framework for selecting the location

Table 1
Comparison of data-driven decision frameworks.

Aspects	Approaches				
	[12]	[13]	[14]	[15]	Our Approach
			Prediction task		
Dataset	Electricity Consumption & Demographics	Not Applicable	Customer GPS information	Smart Home device data	Historical order delivery data & Area-specific location data
Performance Metrics	Does not report	Not Applicable	Recall, Precision, Accuracy	Does not report	Recall, Precision, AUC, G-mean, F1-score, Accuracy
			Schedule generation task		
Outputs Delivery Schedules	Yes	Yes	No	Yes	Yes
Time Required for schedule generation	1 Minute for 15 customers (exponential complexity)	5 h for 100 Customers (exponential complexity)	Not Applicable	Does not report (exponential complexity)	18 min for 4015 and 10 min for 2136 customers (polynomial complexity)
Total number of customers in Simulation	15 (Synthetic locations)	100 (Synthetic locations)	Not Applicable	128 (Real-world locations)	Hub A: 4015 & Hub B: 2136 (Real-world locations)
Simulation Area	324 sq. km.	Not Specified	Not Applicable	5 sq. km.	Hub A: 36 sq. km & Hub B: 156 sq. km

of add-on retail products. Bertsimas & Kallus [38] concatenate features from the company's internal sales & inventory records, IMDb, Rotten Tomatoes, and Google trends to predict sales for inventory management. Similar to these studies, we perform feature level fusion while combining order delivery data and location data.

3. Proposed decision support framework

The proposed framework is intended to generate data-driven delivery schedules that could reduce failed deliveries and delivery costs. Fig. 1 illustrates our predictive and prescriptive analytics framework. We train several machine learning models and employ the best performing model in our framework. As a part of the training, data from two sources (historical order delivery data and location data) are utilized. The location data is captured from an external source; hence latitude and longitude coordinates are used for lookup. The models are trained to predict the chance of delivery success.

Once the best performing model is selected, our approach works in two steps. In the first step, order-level features and location features of various orders are passed to the predictive model, which generates the chance of delivery success for every time point in the delivery shift. These mappings of the chance of delivery success to time points in a day are called order success profiles. In the second step, time periods with the highest delivery success probabilities are identified from these profiles, and delivery slots are inferred. The inferred delivery slots are assigned to respective orders, and through a VRPTW optimization setup, delivery schedules are generated.

The proposed decision support framework is intended to cater to hubs that deliver orders from a few hundred to a few thousand every day. The computational efficiency of the schedule generation procedure is of paramount importance. The more run time the procedure takes, the more it reduces the available time in a delivery shift. Therefore, we prefer models and techniques that require less computational resources.

3.1. Step I: Delivery success prediction

The main objective of the delivery success prediction task is to predict whether the delivery attempt of an order will result in success or failure, using the order-level features, attempt time, and location features as the predictors. Therefore, the dependent variable that we predict is delivery success, and the prediction task is a binary classification. The delivery success variable takes the values 0 and 1, indicating delivery failure or success, respectively. The predictor variables include order and location characteristics along with attempt time. Table 2 lists the predictor variables included from the platform order delivery data.

The variation in location characteristics is captured using amenity counts. We consider 12 common amenities, namely, restaurants, coffee

Table 2
List of predictor variables included from platform order delivery data.

Variable name	Type	Explanation
Category	Categorical	Category of the product. 14 high-level categories are considered
Attempt_Time	Numeric	The time of attempting delivery of an order (in minutes)
Delay	Numeric	The number of days an order is delayed by
Payment_Type	Categorical	The type of payment. 2 categories of payment are considered
Service_Tier	Categorical	The delivery tier of order (e.g., one-day delivery). 4 levels are considered
Weekday	Categorical	Day of delivery. 7 days are considered
Pretermission_1	Numeric	Expected Delivery Date - Order Date (in days)
Pretermission_2	Numeric	Delivery Date - Order Date (in days)
Value	Numeric	The value of the order

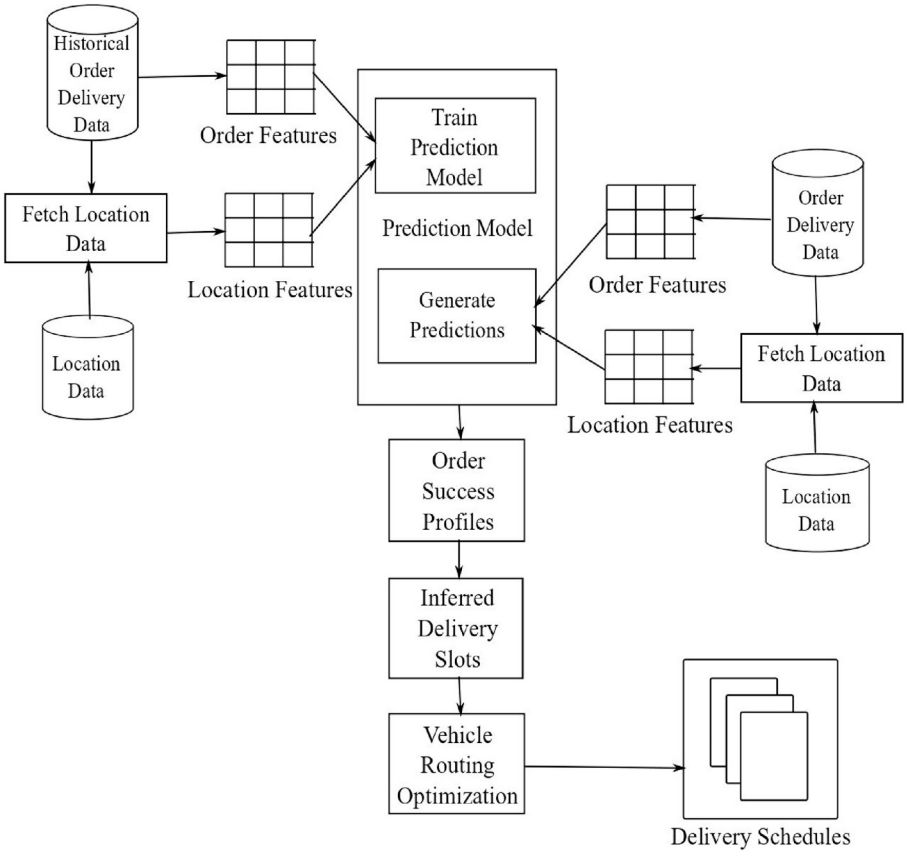

Fig. 1. Illustration of the proposed decision support framework.

shops, fast food outlets, cinema halls, sights & museums, hotels, shopping malls, administrative regions, hospitals, fuel stations, parks and, ATMs. The definitions are listed in Table 3. The counts of these 12 amenities operationalize the location characteristics. For every order destination, the number of amenities present in a radius range of 100, 200, 300, 400, 500, 750, 1000, 1500, 2000 is computed. Therefore, for every amenity, nine different counts are obtained. In total, 108 variables are obtained to represent location characteristics. However, including all 108 variables would have been infeasible due to the problems posed by high dimensionality. Hence, for every amenity, we compress the corresponding nine counts into one variable. We employ robust autoencoders (RAEs) [39] to perform this compression. The features compressed by RAEs are found to be very effective in dimensionality reduction [40].

3.1.1. Learning to predict delivery success

The problem of predicting whether an order will be successfully delivered (delivery success = 1) or not (delivery success = 0) is a binary classification problem. Many of the available machine learning algorithms can solve this task. However, none can be guaranteed to perform the best [41]. Therefore, we select the most popular methods known to yield highly accurate models – Random Forests [42], Gradient Tree Boosting (XGBoost) [43], LogitBoost [44], Artificial Neural Networks [45], and Decision Trees [46]. The first three methods are ensemble methods. Ensemble methods are found to outperform single classifiers on different datasets [47]. Also, ensemble methods are robust [48] and yield better generalization performance [49]. The fourth method, artificial neural networks (ANNs), is becoming the industry choice for predictive analytics [50]. Decision Trees (CART) serves as a baseline. In our study, we employ Scikit-learn [51] for implementing Random Forests, Decision Trees and LogitBoost, XGBoost [43] for implementing gradient tree boosting, and TensorFlow [52] for implementing ANNs.

3.1.2. Imbalanced learning

The target variable for the prediction task (the delivery success variable) is skewed towards successes in our problem. Generally, the delivery successes will be higher than the delivery failures resulting in class imbalance. Class imbalance biases machine learning algorithms towards the majority class and reduces the prediction accuracy of the minority class [53]. To increase the classification performance, we employ several sampling methods and a cost-sensitive learning framework. Specifically, we consider highly performing methods such as weighted loss, random oversampling, random undersampling, and several other variants based on them- SMOTE, Borderline SMOTE, ENN, RENN, SMOTE+ENN, SMOTE+Tomek. Though these methods are found to give good results on imbalanced datasets [30,54–56], their performance varies widely depending on the dataset and classifier algorithm. For instance, several studies [30,54,55] show the wide variation in performance of these methods on several datasets. Since the first aim of the article is to establish the predictive power of the proposed predictors, applying several methods such as these helps us establish the maximum potential of the predictors.

In the cost-sensitive framework (or weighted loss), the cost of misclassifying a minority example is set higher than the cost of misclassifying a majority example. This makes the learning algorithm make fewer mistakes on minority class and improves its minority class accuracy. In our cost-sensitive models, the weights are set inversely proportional to the respective class frequencies while training.

Sampling methods are of two kinds: oversampling and undersampling [53]. In oversampling, a set of minority samples is selected and duplicated until the desired balance is achieved. While oversampling inflates the dataset, undersampling deflates it by selecting and removing the majority class samples. Oversampling suffers from overfitting, and undersampling fails to learn all concepts [55]. We use both of these methods as baselines. SMOTE, Borderline SMOTE, ENN, RENN, SMOTE+ENN, SMOTE+Tomek are proposed to overcome the drawbacks of oversampling and undersampling. The first two methods are based on oversampling; the following two methods are based on undersampling and, the last two methods are based on both oversampling and undersampling. For further details on imbalanced learning, readers can consult Branco, Torgo, & Ribeiro [57].

3.1.3. Metrics

As discussed in the earlier section, our classification task belongs to an imbalanced learning problem. Traditional metrics such as accuracy and error rate can be deceiving in such tasks. Therefore, for evaluating models, we use metrics such as G-mean and AUC, which are widely used for imbalanced learning tasks [53]. G-mean is the geometric mean of sensitivity and specificity. AUC is the area under the receiver operating characteristic curve (ROC). Both metrics give equal importance to both classes. We also report other prevalent measures such as recall, precision, accuracy, F1-score, and confusion matrices to complement these metrics.

3.1.4. Order success profiles

The developed machine learning models are used to generate order success profiles (OSP). Order success profiles map the probability of delivery success to the time points in a day. Each order is associated with such a profile. Profiles help us find the time slots during which order delivery is likely to result in success.

The machine learning models discussed in the previous section estimate the probability of delivery success as a function of order characteristics and location characteristics. For an order, throughout the day, all the features except the attempt time remain constant. Therefore, by predicting the probability of delivery success at different points in time, keeping all other features constant, we can map the chance of delivery success to time. These mappings provide valuable information about an order's delivery success with respect to time. For instance, Fig. 2 shows different order success profiles. In these profiles, the horizontal axis represents the time in hours, and the vertical axis represents the probability of delivery success. The title of the profile represents the order id. For orders corresponding to these profiles, the chance of delivery success drops below the threshold (of 0.5) after a point in time indicated by the vertical dashed line; for AID_353, it is about 9:50 h, similarly, for AID_270, it is about 14:00 h, and for AID_172 it is about 17:50 h. Therefore, if these three orders are to be delivered, the sequence should be AID_353, AID_270, AID_172. Similarly, every order can be associated with a unique profile that can guide the schedule generation.

Table 3
Location Predictors.

Variable name	Explanation	Variable name	Explanation
Fuel	Number of Fuel Stations	Sights & Museums	Number of tourist attractions, museums, monuments
Coffee	Number of Places where coffee, tea, other drinks & refreshments are served	Leisure	Number of recreational facilities: sports fields, parks, gardens, zoos
Admin	Number of named squares, road intersections, office buildings	Transport	Number of train stations, bus stations, bus stops, taxi stands
Health	Number of hospitals and other healthcare facilities	Out	Number of entertainment spots: pubs, clubs, cinema halls
Restaurant	Number of places where meals are served	Snacks	Number of fast-food spots, street food stalls
Shopping	Number of shopping malls, furniture stores, apparel stores	ATM	Number of ATMs

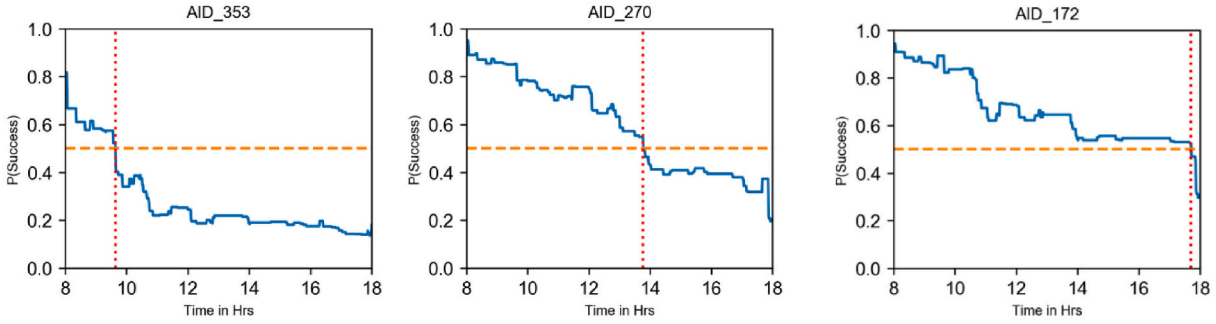

Fig. 2. Order success profiles (OSPs).

3.2. Step II: Delivery schedule optimization

Delivery schedules define the sequence of order deliveries. They also specify the time windows during which orders should be delivered. Solving the VRPTW problem generates the required delivery schedules. In the VRPTW framework, each customer i is associated with a preferred delivery slot (E_i, L_i), the delivery of the order or service should strictly occur between E_i and L_i. In our problem, preferred delivery slots are replaced with the time windows generated from order success profiles. From the order success profiles, time intervals during which the delivery is predicted to be successful are identified and treated as delivery slot requirements. For example, observe Fig. 2, order AID_270's chance of delivery success is above the threshold between 8:00 to 14:00 h; therefore, (8,14) or (480,840) (expressed in minutes) is treated as a delivery slot.

Our schedule development problem has two unique traits. First, each order can be associated with multiple time slots of differing lengths. Therefore, each order i is associated with a set of delivery slots $\{(E_{i1}, L_{i1}), (E_{i2}, L_{i2}), \ldots\}$. This has implications for schedule generation. While constructing schedules, inserting orders that have a relatively smaller and fewer number of time slots before inserting the others makes schedule generation flexible and reduces the number of schedules generated. Accordingly, we assign a measure called priority to each order. The priority of an order can be calculated as follows:

Priority(p) = 1 – (minutes available / size of delivery period).

Here "minutes available" is the number of minutes for which an order is predicted to be delivered successfully. It is calculated by summing up all the time slots' lengths associated with an order. "Size of delivery period" is the total duration of an agent's delivery shift, typically 540 min (9 h). The priority measure distinguishes between orders, such that orders with large durations of success have low priority while orders with small durations have high priority. We use the priority of orders while generating the schedules.

The second trait is derived from a practical requirement. From our conversations with the logistics managers, we learned that the delivery agents who are being assigned to the same locations, every day, are more efficient than others. Delivery agents gain efficiencies because delivering orders in the same area allows them to learn route characteristics and navigate better in the succeeding trips. Accordingly, we consider specific directions around the hub along which orders are clustered and included in the schedule.

Solving the schedule generation problem requires the addition of new constraints to the traditional VRPTW problem, which is itself an NP-hard problem [58]. Therefore, we employ heuristic techniques to generate initial schedules. Specifically, we adapt the insertion heuristic [59] by including the two discussed traits. The schedules are further improved using iterated local search.

3.2.1. Schedule generation

We define the schedule generation algorithm in this section. We consider a set of n orders. Each order is associated with a set of feasible time slots $\{(E_{i1}, L_{i1}), (E_{i2}, L_{i2}), \ldots\}$, and a priority measure p_i. The delivery hub is located at (h_{lon}, h_{lat}), where $\ddot{h}_{lon}$ is the longitude, and $\ddot{h}_{lat}$ is the latitude. Let there be R empty schedules. Each schedule r_j is associated with a direction θ_j, span Φ_j. Both the span and direction are angles measured in degrees. Any schedule r_j can only serve the orders located between $[\theta_j - \Phi_j, \theta_j + \Phi_j]$. We assume a homogeneous fleet of vehicles where each vehicle has a capacity Q. Algorithm 1, which is loosely based on the insertion heuristic [59], is used to generate the initial schedules. Note that a schedule is represented as a list of orders. The first and the last entry in the list is the hub itself. While inserting a new order into the schedule, schedule capacity, region to which the order belongs, and the time feasibility are checked. Time feasibility is verified using the procedure given by Campbell & Savelsbergh [59]. The feasible position, which costs the least in terms of distance, is chosen as the insertion position. If none of the available positions are feasible, the order is added to a new schedule that is directed towards the order's location with default span, $\Phi_{default}$. The generated schedules using Algorithm 1 are improved using iterated local search which performs exchanges based on 2-opt* heuristic [60].

Algorithm 1
Procedure to generate schedules

```
N = list of orders sorted in priority descending order
for order ϵ N
  set cost* to a very high value (e.g., 999999999999999)
  for r ϵ R
    if same_region(order, r) and capacity_exists(order, r)
      for (prev, next) ϵ r
        for slot in order.slot_options
          if time_feasible(prev, order, next) and costs(prev, order, next) < cost*
            prev* = prev
            next* = next
            slot* = slot
            cost* = costs(order, prev, next)
            r* = r
  if slot* is not set
    r* = create_new_route(order.θ, Φ_default)
    (prev*, next*) ϵ r
    slot* = order.slot_options[1]
    add(r*,R)
  insert_order(order, prev*, next*, slot*, r*)
return R
```

4. Experimental setting

4.1. Case study

The proposed framework is intended to generate data-driven delivery schedules that could reduce failed deliveries and delivery costs. For this study, we consider an e-commerce delivery hub that fulfills customers' orders in a 9-h delivery shift. As a part of the fulfillment process, the delivery hub receives orders from the warehouse in the

early hours; orders with premium service levels (1-day, 2-day deliveries) may arrive throughout the day. The delivery manager generates delivery routes or schedules once in the morning and the afternoon, optimizing for travel distance. The majority of the orders are covered by the delivery shift that starts in the morning. These delivery schedules are then assigned to delivery agents who deliver the respective orders to customers. If a customer is not available to receive the shipment, agents proceed to the next customer, and the respective shipment is returned to the hub at the end of the day. The returned shipment is considered a failed delivery (or a delivery failure) and reattempted the next day. Each order is attempted a maximum of three times before a successful delivery. If none of the attempts are successful, then the respective order is returned to the warehouse. Currently, delivery schedules are generated without considering appropriate times for order deliveries.

4.2. Data

For experimental validation, we use order delivery data provided by one of the leading e-commerce platforms in India and extract amenity counts from Here Technologies' location platform [20] as described in Section 3.1. Based on the discussions with our industry collaborator, we chose two diverse delivery hubs A and B. The two hubs are located 2000 km apart and are diverse both in terms of geography and economy. Therefore, the distribution of predictor variables between the hubs is varied. For instance, 5% of Hub-A's orders and 48% of Hub-B's orders employ economy-delivery. Similarly, there are differences in payment preferences, popular product categories, and average order value. Also, there is a significant difference in the distribution of amenities. For instance, shopping spots are densely distributed around Hub—B, while parks, zoos, and sports fields are densely distributed around Hub-A. Besides, Hub-A is located in a major city that serves an area of 36 sq. km, while Hub-B is located in a town serving a relatively large 156 sq. km area. The variation between the hubs helps us inquire the predictive power of the proposed models and the generalizability of the proposed decision framework.

For Hub-A, the industrial collaborator supplied 11 weeks of order delivery data. The data cleaning involved the removal of records that have NULL values in the fields of interest. Around 6% of the records were removed. The first ten weeks' data is used for model development, while the last week's data is used for model evaluation. After data cleaning, 81,967 order deliveries became available for training and 4015 order deliveries for testing. About 11% of the orders are delivery failures in the whole dataset. Orders in the test data are used to generate delivery schedules for both the baseline and proposed approaches in simulation experiments.

For Hub—B, 13 weeks of order delivery data were available. Similar to Hub-A's case, the data cleaning process involved the removal of records with NULL values in the fields of interest. Around 9% of the records were removed. Since Hub-B is located in a town, the number of order deliveries were less than half of the Hub-A's volume. Therefore, to build accurate models, the whole town's data for the first 12 weeks is used to train the predictive models. Though the models are trained on the whole town dataset, since they are supposed to be deployed at the hub, we test the models and run simulation experiments on the last week's data corresponding to the hub. This approach of training predictive models with the whole town dataset can be used for hubs located in small towns where the amount of delivery data becomes insufficient to build accurate models. Finally, 84,714 records became available for training and 2136 records for testing. About 17.9% of Hub-B's delivery attempts resulted in failures.

Fig. 3 shows the distribution of orders. The blue dots represent successful deliveries, while the orange dots represent delivery failures. The delivery hub is indicated by the square colored in magenta. In both locations, delivery failures are unevenly distributed; some areas are more prone to failures than others.

4.3. Experimental setup for building predictive models

The hyperparameters of the machine learning models of respective hubs are determined by performing a grid search. For each algorithm and imbalanced learning method combination, hold-out cross-validation is performed where the last week's data from the training set is used as a validation set. The models are trained to maximize the AUC score. Once the best performing hyperparameters are determined, every model is retrained with the whole training set using the determined parameters and tested on the respective test dataset. We perform two statistical tests to verify whether the performance differences between classifiers are statistically significant. We follow the procedure recommended by

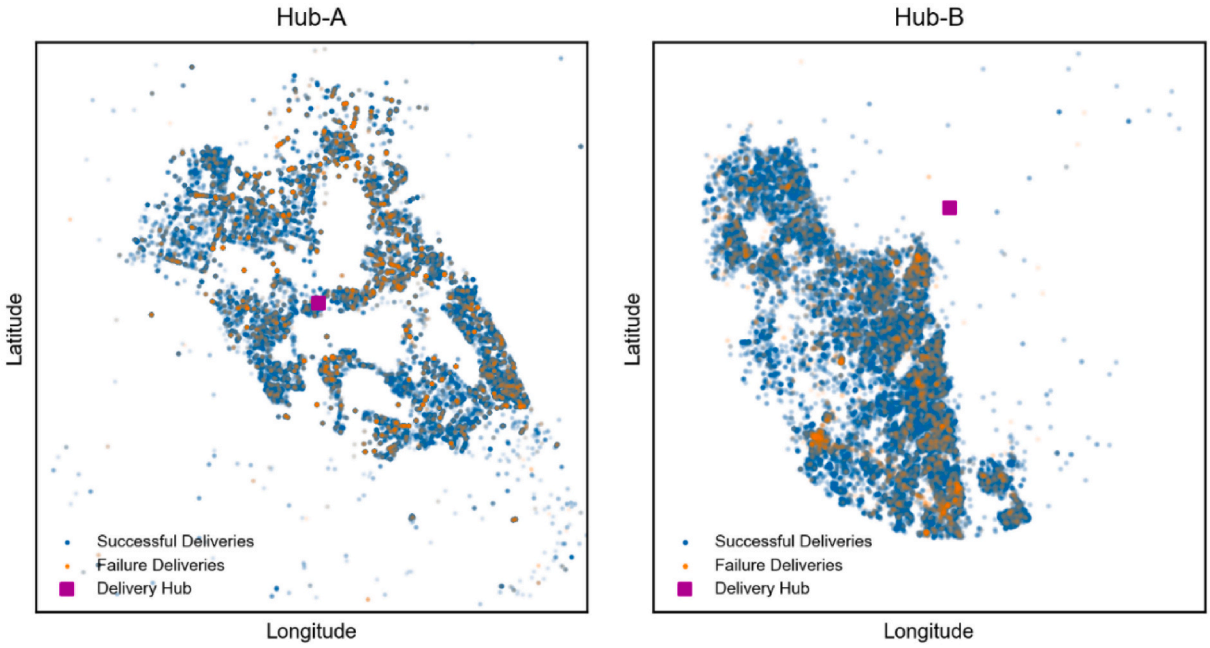

Fig. 3. Spatial distribution of orders corresponding to Hub-A and Hub—B.

Japkowicz & Shah [61] and employ Friedman omnibus test and Nemenyi post hoc tests at a significance level of 0.01.

For Random Forests, the number of trees is set to 3000 while tuning for the maximum number of features in an individual tree, the minimum size of a leaf node, and the minimum number of samples required for splitting an internal node. For XGBoost, step size shrinkage, minimum loss reduction required to split a node, maximum depth of a tree, subsample ratio, and columns used to construct a tree are tuned. For LogitBoost, the learning rate and the number of estimators are tuned. For Artificial Neural Networks, the number of layers, hidden units, L2-regularization coefficient, batch size, epochs are tuned with ReLU activation. For decision trees, Gini impurity is used to measure the node's quality while tuning for maximum depth of the tree, minimum samples required to split a node, and minimum samples in a leaf.

4.4. Experimental setup for testing the two-step approach

To evaluate the performance of our approach, we simulate two policies-baseline policy and data-driven policy. The baseline policy reflects the current industry practice, while the data-driven policy utilizes our approach. In both the policies, we simulate the order delivery process on seven days of test data corresponding to Hub-A and Hub—B, respectively.

Baseline policy is the current industry practice where no prediction model is available, and the generated schedules are optimized to reduce travel distance. Since no information on appropriate time windows is available here, we do not enforce any time slot constraints during the simulation. In other words, for every order, we assign the longest possible delivery slot (420 min, 960 min) or (7 AM, 4 PM), which is the 9-h delivery shift itself. From a scheduling perspective, since all orders can be delivered throughout the delivery shift, the travel distance is minimized while generating the schedules. This setup resembles VRP optimization (that is, VRPTW without time slot constraints). However, since the orders are attempted without considering appropriate time slots for delivery, some of them are bound to result in failed deliveries. We assume that the delivery failures recorded in the test data occur. During the simulation, every day after the delivery process, x% of orders are assumed to be failed deliveries and are returned to the hub. The returned orders are attempted the next day. These x% of orders are selected randomly. For Hub-A, 12.61% of orders in test data resulted in delivery failures; hence, x is set to 12.61. For Hub—B, the recorded failure rate is 15.5% in test data; hence x is set to 15.5%.

In the data-driven policy, the prediction model is employed; therefore, order success profiles and order delivery slots are available. If an order is delivered in the inferred delivery slot, it is assumed to be a delivery success. However, since the precision of the predictive models is not 100%, a few time points that have a relatively low probability of delivery success will also be included in the inferred delivery slots. Therefore, we reduce the width of delivery slots, depending on the precision of our predictive models. For instance, consider an order whose inferred delivery slot is (540, 600). Let the precision of the predictive model be 94%, out of 60 min of the delivery slot, 6% of the minutes, or 3.6 min are false positives (minutes that may not lead to delivery success). Hence, we adjust the delivery slot width removing 3.6 min from the inferred slot on either side of the slot. In this case, 2 min (rounded up from 1.8 min) can be removed from either side. Therefore, (542, 598) becomes the new adjusted delivery slot. This procedure is followed for adjusting all delivery slots in the data-driven policy. Also, all orders with delivery slot widths smaller than 30 min are considered as failures and are assigned with a delivery slot (420, 960). The failed orders are rescheduled the next day, similar to the baseline policy.

In the simulations, haversine distance [62] with a rescaling factor of 1.5 is used for computing distance. 16kmph and 20kmph are assumed to be the average speed of vehicles around Hub-A and Hub—B, respectively. The vehicle capacity is 50 orders. For both the hubs, 6 initial directions with $\theta = \{0, 60, 120, 180, 240, 300\}$ are considered. The span of each schedule is set to 45 degrees. The daily cost of operating a vehicle is 633 INR, and the fuel costs are 2.7 INR/km. The delivery shift starts at 420 min (7 AM) and ends at 960 min (4 PM). The simulations are run for 100 times for both the hubs, and on average, the schedule generation took 18 min for Hub-A and 10 min for Hub—B.

5. Results

5.1. Results for the delivery success prediction

Tables 4, 5, 6, 7, 8 summarize the predictive modeling results. The first two tables report the AUC scores, from which we observe that the scores are reasonable and similar for all the models. This result suggests that models have good average performance in discriminating between delivery success and failure. However, to identify the best classifier, we employ statistical tests. For both hubs A & B, the observed differences between various classifiers are significantly different, indicating that all of them do not have equivalent performance. To identify the exact pairs of classifiers which are significantly different, we perform post hoc tests as described in Section 4.3. The differences between all the pairs of classifiers are significant except the pair of Random Forests and XGBoost, which happen to have high AUC scores. Therefore, we choose Random Forest and XGBoost classifiers and gauge their exact performance at a particular threshold. For this study, we choose a threshold of 0.5, which offers equal priority to both classes. The proposed framework utilizes the selected models extensively; therefore, we prefer an algorithm and imbalanced learning scheme that requires less computational resources for training and prediction for a faster real-world response.

Table 6 reports the G-mean scores of Random Forests and XGBoost classifiers under different imbalance correction methods tested on data corresponding to hubs A & B, respectively. The difference in performance between the classifiers for both the hubs is insignificant; therefore, none have a clear advantage. Out of the two, we choose the XGBoost classifier because of its superior computational performance. For instance, XGBoost took 4.4 milliseconds for prediction on the test data, while Random Forest took 2.58 seconds. Also, the trained XGBoost model occupied 0.59 MB of space while Random Forests occupied 712.42 MB. The prediction speed and low memory footprint of XGBoost make it the ideal choice for our application.

When all imbalanced learning methods are compared, the differences between them are found to be significant. Therefore, all of them do not have equivalent performance. Across the highest performing methods, for Hub-A, the differences between random oversampling, random undersampling, and weighted loss methods are insignificant. For Hub—B, the differences between RENN, random oversampling, random undersampling, and the weighted loss methods are insignificant. Out of these methods, we chose the weighted loss method because of the ease of training and consistency. Random Oversampling inflates the data set and requires additional computational effort for training. Random Undersampling may eliminate useful examples and decrease the performance of future predictions. RENN has significantly low

Table 4
AUC scores of models trained on Hub-A's dataset.

Imbalanced learning method	Models				
	RF	XGB	LB	ANN	CART
SMOTE	**72.42**	71.51	70.78	69.54	70
SMOTE-Borderline	**72.56**	71.67	70.91	70.34	69.63
SMOTE+ENN	**72.78**	72.47	71	71.73	71.17
SMOTE+Tomek	**72.50**	72.40	70.79	70.36	70.03
ENN	73.36	**73.47**	71.09	72.93	71.8
RENN	73.16	**73.63**	71.02	72.43	71.35
Oversampling	73.36	**73.61**	70.93	73.36	71.15
Undersampling	72.65	**72.85**	71.18	72.79	71.22
Weighted Loss (Cost-Sensitive Framework)	73.2	**73.65**	64.83	72.06	71.56

Table 5
AUC scores of models trained on Hub-B's dataset.

Imbalanced learning method	Models				
	RF	XGB	LB	ANN	CART
SMOTE	**78.43**	78.31	76.89	77.09	75.11
SMOTE-Borderline	77.7	**78.76**	76.98	77.17	75.24
SMOTE+ENN	**78.57**	78.28	76.63	77.32	75.4
SMOTE+Tomek	**78.71**	78.67	76.89	77.51	75.25
ENN	78.84	**79.01**	76.85	77.12	75.56
RENN	78.35	**78.38**	77.02	77	76.71
Oversampling	78.37	**78.97**	77.08	76.96	76.07
Undersampling	78.98	**79.08**	77.23	77.64	75.83
Weighted Loss (Cost-Sensitive Framework)	**79.12**	78.77	70.81	77.22	76.13

RF stands for Random Forests, XGB stands for XGBoost, LB stands for LogitBoost, ANN stands for Artificial Neural Network, and CART represents Decision Trees. The highest AUC values under each imbalanced learning method are shown in bold.

performance for Hub-A. Therefore, we chose XGBoost with the weighted loss for both the hubs.

Tables 7, 8 show the confusion matrices of the selected XGBoost classifiers evaluated on test datasets. For Hub-A, from Table 7, we observe that the classifier identifies 9.66% of delivery failures out of 12.61%. Also, the model has high precision, 94.46%, in identifying successful deliveries. For Hub—B, from Table 8, we observe that the classifier detects reasonable amounts of failed deliveries, 12.31% out of 15.49%, with very high precision, 94.87%. Our classifiers have better predictive power than similar models proposed in the literature. The model proposed by Praet & Martens [14] has a recall, precision, and f1-score of 75%, 41%, and 53%, respectively, while the models we develop have a recall, precision, and f1-score of 58%, 94%, and 72% for Hub-A and 70%, 95% and 80% for Hub-B respectively.

5.2. Results for the two-step framework

The results of the simulation experiments are presented in Fig. 4 and Table 9. We compare the performance of data-driven policy against baseline policy in terms of attempts required, distance traveled, vehicles needed, and the costs of delivering orders. Simulation results support the usage of the data-driven policy over the baseline policy that is currently in practice. Utilizing our approach not only results in lesser delivery costs but also minimizes package handling and storage costs that result due to failed deliveries.

Fig. 4 shows the number of delivery attempts required for seven days of test data for both the policies considered. The "New Deliveries" indicated by the green bar represents the actual new deliveries scheduled for that day; these order attempts are the first-time delivery attempts. The difference in length between bars corresponds to the failed deliveries of previous days. On the first day of delivery, the number of order deliveries is the same for both the policies. However, during the subsequent days, implementing data-driven policy results in fewer order deliveries relative to the baseline policy. This is because baseline policy results in relatively more order delivery failures that are reattempted the next day, increasing the total number of delivery attempts. Also, the number of attempts under the data-driven policy is slightly higher than the new delivery attempts indicating a lower delivery failure rate than the baseline policy. The increased delivery volume demands more vehicles and results in additional package handling. The overall increase in vehicles increases the capital and labor costs significantly. Table 9 shows the total vehicle and distance costs.

Table 9 reports the performance of both the policies when applied to hubs A & B. The data-driven policy incurs fewer travel costs for both the hubs. In the baseline policy, the number of failed deliveries is higher, which increases the total number of attempts; therefore, the total

Table 7
XGBoost confusion matrix evaluated on Hub-A's test data (in %).

	Predicted Values	
Actual Values	Success	Failure
Success	50.39	37.00
Failure	2.95	9.66
Recall: 58%, Precision: 94%, Accuracy: 60%		

*XGBoost Parameters: (step size shrinkage: 0.05, minimum loss reduction: 0.1, maximum depth: 7, subsample ratio: 0.7, column sample ratio: 0.675).

Table 8
XGBoost confusion matrix evaluated on Hub-B's test data in (%).

	Predicted Values	
Actual Values	Success	Failure
Success	58.90	25.61
Failure	3.18	12.31
Recall: 70%, Precision: 95%, Accuracy: 71%		

*XGBoost Parameters: (step size shrinkage: 0.05, minimum loss reduction: 0.2, maximum depth: 7, subsample ratio: 0.4, column sample ratio: 0.23).

Table 6
G-mean scores of models tested on Hubs A & B.

		Imbalance Learning Method								
Classifier	Metric	SMOTE	SMOTE-Borderline	SMOTE+ENN	SMOTE+Tomek	ENN	RENN	O.S.	U.S.	Weighted Loss
					Hub-A					
RF	G-mean	**36.44**	**40.24**	**60.29**	**38.95**	41.55	50.03	66.68	66.07	64.15
	Recall	96.01	94.18	78.93	94.78	94.95	90.48	65.80	55.51	74.12
	Precision	88.54	88.74	91.02	88.66	88.94	89.66	93.36	94.74	92.03
XGB	G-mean	34.33	35.90	55.52	33.28	**44.23**	**52.81**	**66.73**	**66.11**	**66.48**
	Recall	96.23	95.95	83.87	96.63	94.30	88.77	63.12	56.56	57.65
	Precision	88.38	88.48	90.19	88.33	89.19	89.97	93.69	94.53	94.46
					Hub-B					
RF	G-mean	**58.15**	**61.94**	**70.16**	**62.1**	60.13	68.54	**70.35**	71.02	68.79
	Recall	92.01	88.62	65.77	90.04	91.66	80.59	71.03	68.63	78.17
	Precision	89.26	89.93	93.73	90.00	89.64	91.70	93.05	93.67	91.88
XGB	G-mean	51.47	50.05	69.14	48.98	**62.64**	**68.9**	69.49	**71.37**	**73.83**
	Recall	95.14	95.36	78.82	95.85	89.47	80.22	77.60	69.61	69.69
	Precision	88.29	88.08	91.97	87.96	90.11	91.83	92.15	93.68	94.87

Note: RF stands for Random Forests, XGB stands for XGBoost. O.S. stands for Oversampling, U.S. stands for Undersampling. The highest G-mean score under each imbalanced learning method for each hub is shown in bold.

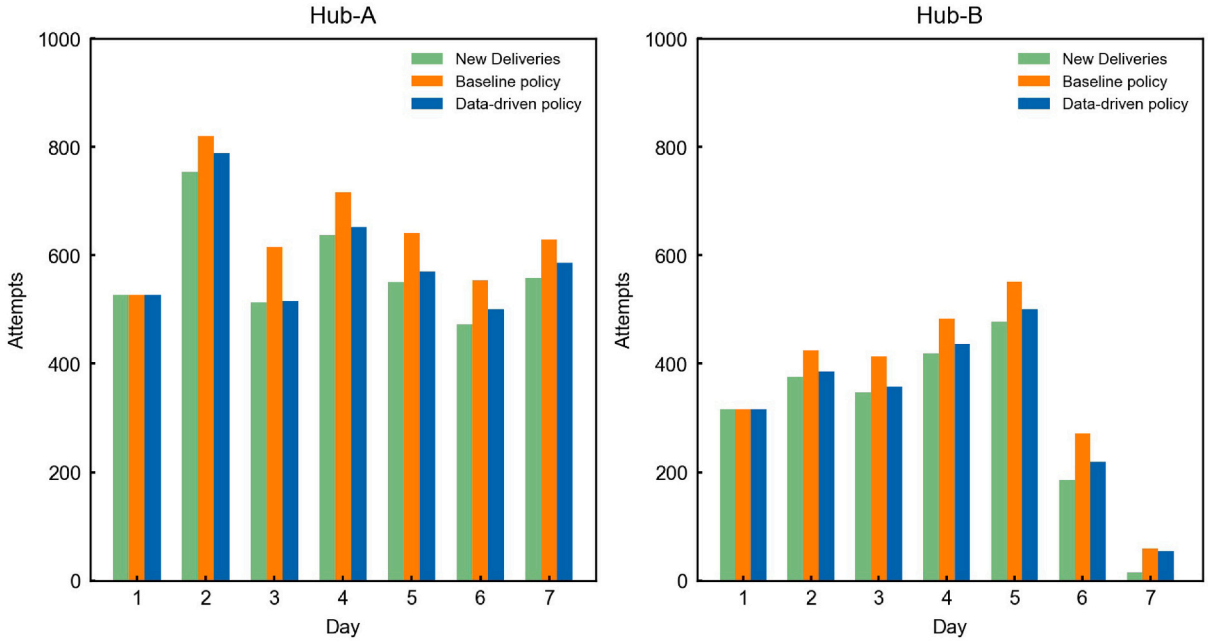

Fig. 4. Comparison of order attempts per day between baseline and data-driven policies of hubs A & B.

Table 9
Performance of baseline policy and data-driven policy of hubs A & B.

Policy	Hub	Total orders	Total attempts	Total vehicles	Total distance (km)[a]	Total costs (INR)[a]	Std Dev (Total costs)[a]	Savings (%)[a]
Baseline	A	4015	4503	112	1052.23	73,737	24.05	–
Data-driven		4015	4142	103	1178.55	68,381	19.14	7.2
Baseline	B	2136	2518	64	1315.28	44,063	40.74	–
Data-driven		2136	2270	57	1280.67	39,538	48.03	10.2

[a] The "Total Distance" and "Total Costs" are the average values of respective quantities across 100 simulation experiments. The "Std Dev" is the standard deviation of total costs across 100 simulation experiments. "Savings" is the percentage difference in total costs between respective policies vs. baseline policy for the given hub.

number of vehicles required is higher. In contrast, data-driven policy results in fewer delivery failures resulting in fewer attempts. However, the total distance traveled may be higher for the data-driven policy, as in the case of Hub-A. This increase in distance is due to the applied time window constraints. Though the distance traveled may be higher when implementing the data-driven policy, since vehicle and labor costs are far higher than fuel costs, the total costs is lower for the data-driven policy. On average, delivering an order is cheaper with the data-driven policy for both the hubs. The resultant savings of 7.2% and 10.2% are significant and could drive down the cost of delivery per unit. Besides, package handling and storage costs can also be reduced.

6. Discussion and conclusion

Predicting order delivery success is beneficial for e-commerce companies. Companies can plan deliveries accordingly, thereby increase successful deliveries. Though prior attempts were made, as discussed in Section 2, the approaches require sensitive customer-specific information that is not widely available. In this paper, we proposed a predictive and prescriptive analytics framework that schedules order deliveries in the most appropriate time periods. During the development, we generated novel features and explored several machine learning algorithms. XGBoost classifier trained under a cost-sensitive scheme gave the best model performance while consuming less computational resources. The predictive models we built have significantly better performance than related models proposed in the literature.

Our approach is applied to two real-world order delivery datasets, which were supplied by one of the prominent e-commerce firms in India. The developed predictive models were able to detect a significant amount of delivery failures; 9.66% of delivery failures out of 12.61% for Hub-A and 12.31% of delivery failures out of 15.5% for Hub-B. Predictive models such as these can identify orders which are less likely to be delivered and allow the platforms to take up remedial measures to avoid wasted delivery attempts. We also demonstrated the usage of predictive models in generating delivery schedules. Our simulation experiments indicate that the schedules generated using our framework can result in cost savings of 7.2% and 10.2% for hubs A & B, respectively. The savings were relatively more significant for Hub-B, where delivery failures were the highest. This shows that platforms can gain significantly by applying the proposed approach to areas that are more prone to delivery failures.

Our study informs managers in several ways. As discussed, failed deliveries are one of the critical problems of e-commerce last mile, and failure rates of 25% are common [19]. Therefore, controlling delivery failures can significantly reduce delivery costs. One of this study's messages is to leverage historical order delivery data and amenity counts in estimating delivery success. These sources of data are not exclusive to any platform; hence platforms can leverage the proposed approach. Second, the study shows that models trained on the whole city data (comprising of many individual hubs) perform well on the subset of data corresponding to a single hub. Following this observation, platforms can build one model for a city instead of a separate model for every hub; this modeling approach reduces the total effort required to train, test, integrate, deploy and maintain predictive models. Third, the study shows that the proposed data-driven policy results in cost savings compared to the baseline policy currently in practice. The resultant cost savings are due to the reduction in the number of attempts and the number of vehicles. It is essential to observe that the number of vehicles can increase

if vehicles' effective capacity reduces. Therefore, cost-savings depend on the number of customers a vehicle can serve, which further depends on traffic and service area. Traffic slows down vehicular movement and limits the number of customers a vehicle can serve. Therefore, cost-savings decrease with an increase in traffic. As the service area increases, the vehicle must cover more distance, which affects the number of customers it can reach. Under low traffic conditions, the influence is negligible but with high traffic, servicing larger areas requires more vehicles. Nevertheless, in our simulations, we used the lowest observed speed for both the hubs. Therefore, the actual speed and the cost savings can be higher. Finally, the proposed approach demonstrates that the delivery service level, in terms of delivery promptness, can be increased without increasing the delivery fleet size.

Our work can be expanded on two fronts. On the predictive modeling side, novel features can be identified or engineered to improve prediction performance. As the performance increases, more accurate schedule prescriptions can be made. For instance, improving the sensitivity of models increases the models' ability to identify more successful time points, thereby resulting in longer time slots. Longer time slots reduce the constraints on VRPTW optimization resulting in a reduction in the number of vehicles required and the distance traveled. This helps in driving down the delivery costs. Besides, large amounts of data can be used to build models with better generalization performance. In this study, we have used three months of data; whether using more data can improve performance can be explored. Also, the minimum amount of data required to obtain the necessary performance can be explored. Building models with minimum data reduces the time needed to train and test models; this enables platforms to deploy models and achieve efficiency quickly.

On the schedule generation front, we have extended the insertion heuristic. Future work can extend other construction heuristics to the problem setting and identify the best heuristic for scheduling in terms of convergence time and solution quality. Also, novel heuristics that offer high-quality solutions in the lowest possible time can be explored. Faster performance helps decrease schedule generation time. This allows delivery managers to schedule orders that arrive at the last minute. A quicker schedule generation method reduces the planning time and keeps the delivery shift broader. Our approach results in an unequal distribution of orders among delivery agents and may result in workload imbalance. A new method that could result in equal or near-equal distribution of orders can be explored; this balances work-load on agents resulting in improved morale and efficiency.

Credit author statement

The authors do not wish to include any author contribution statement for the paper.

Acknowledgments

This work was supported by the Indo-Dutch Grant [grant number 13 (4)/2018-CC&BT] provided by the Ministry of Electronics & IT (MeitY), India, in collaboration with the Netherlands Organization for Scientific Research (NWO).

We thank Flipkart India Pvt. Ltd. for providing necessary data for the research work. In particular, we thank Chandrasekhar K, Shailendra Kumar, Sreeparna Chatterjee, Muthusamy Chelliah for providing advice and sharing the data.

References

[1] J. Clement, E-commerce Share of Total Global Retail Sales from 2015 to 2023, Statista, 2019. https://www.statista.com/topics/871/online-shopping/ (Accessed June 24, 2020).
[2] P. Mikko, Y. Hannu, H. Jan, Solving the last mile issue: reception box or delivery box? Int. J. Phys. Distrib. Logist. Manag. 31 (2001) 427–439.
[3] R. Gevaers, E. Van de Voorde, T. Vanelslander, Characteristics and typology of last-mile logistics from an innovation perspective in an urban context, in: C. Macharis, S. Melo (Eds.), City Distribution and Urban Freight Transport, Edward Elgar Publishing, Cheltenham, 2011, pp. 56–71.
[4] S. Rao, E. Rabinovich, D. Raju, The role of physical distribution services as determinants of product returns in Internet retailing, J. Oper. Manag. 32 (2014) 295–312.
[5] F. Arnold, I. Cardenas, K. Sörensen, W. Dewulf, Simulation of B2C e-commerce distribution in Antwerp using cargo bikes and delivery points, Eur. Transp. Res. Rev. 10 (2018) 1–13.
[6] A. Bhatnagar, S. Misra, H.R. Rao, On risk, convenience, and internet shopping behavior, Commun. ACM 43 (2000) 98–105.
[7] S. Nagy, E-commerce in Hungary: a market analysis, Theory Methodol. Pract. 12 (2016) 25–32.
[8] T. Vanelslander, L. Deketele, D. Van Hove, Commonly used e-commerce supply chains for fast moving consumer goods: comparison and suggestions for improvement, Int J Log Res Appl 16 (2013) 243–256.
[9] T. Chan, Z. Liu, W. Zhang, Delivery service, customer satisfaction and repurchase: evidence from an online retail platform, SSRN Electron. J. (2018), https://doi.org/10.2139/ssrn.3258106.
[10] H. Buldeo Rai, S. Verlinde, C. Macharis, Unlocking the failed delivery problem? Opportunities and challenges for smart locks from a consumer perspective, Res. Transp. Econ. (2019), https://doi.org/10.1016/j.retrec.2019.100753.
[11] A. Kedia, D. Kusumastuti, A. Nicholson, Acceptability of collection and delivery points from consumers' perspective: a qualitative case study of Christchurch city, Case Stud. Transport Policy. 5 (2017) 587–595.
[12] S. Pan, V. Giannikas, Y. Han, E. Grover-Silva, B. Qiao, Using customer-related data to enhance e-grocery home delivery, Ind. Manag. Data Syst. 117 (2017) 1917–1933.
[13] A.M. Florio, D. Feillet, R.F. Hartl, The delivery problem: optimizing hit rates in e-commerce deliveries, Transp. Res. B Methodol. 117 (2018) 455–472.
[14] S. Praet, D. Martens, Efficient parcel delivery by predicting customers' locations*, Decis. Sci. 51 (2020) 1202–1231.
[15] R. Mangiaracina, A. Perego, A. Seghezzi, A. Tumino, Smart home devices and B2C E-commerce: A way to reduced failed deliveries, in: K. Pawar, A. Potter, H. Rogers, C. Glock (Eds.), International Symosium on Logistics, Centre for Concurrent Enterprise, Nottingham, 2019, pp. 189–196.
[16] S. Chakravarty, Location Data Sharing: What are the Concerns Around Privacy?, Geospatial World. https://www.geospatialworld.net/blogs/is-your-location-data-safe/, 2018 (Accessed February 10, 2021).
[17] Internet Society, The Trust Opportunity: Exploring Consumer Attitudes to the Internet of Things. https://www.internetsociety.org/resources/doc/2019/trust-opportunity-exploring-consumer-attitudes-to-iot/, 2019 (Accessed February 10, 2021).
[18] V. Anant, L. Donchak, J. Kaplan, H. Soller, The Consumer-data Opportunity and the Privacy Imperative, McKinsey & Company, 2020. https://www.mckinsey.com/business-functions/risk/our-insights/the-consumer-data-opportunity-and-the-privacy-imperative (Accessed February 10, 2021).
[19] J.H.R. Van Duin, W. De Goffau, B. Wiegmans, L.A. Tavasszy, M. Saes, Improving home delivery efficiency by using principles of address intelligence for B2C deliveries, Transport. Res. Proc. 12 (2016) 14–25.
[20] Here, Here Places API, Here Developer Documentation. https://developer.here.com/documentation/places/dev_guide/topics_api/resource-search.html, 2020 (Accessed October 11, 2020).
[21] R. Mangiaracina, A. Perego, A. Seghezzi, A. Tumino, Innovative solutions to increase last-mile delivery efficiency in B2C e-commerce: a literature review, Int. J. Phys. Distrib. Logist. Manag. 49 (2019) 901–920.
[22] J. Edwards, A. McKinnon, T. Cherrett, F. McLeod, L. Song, Carbon dioxide benefits of using collection–delivery points for failed home deliveries in the United Kingdom, Transport. Res. Rec.: J. Transport. Res. Board 2191 (2010) 136–143.
[23] A.M. Campbell, M.W.P. Savelsbergh, Decision support for consumer direct grocery initiatives, Transp. Sci. 39 (2005) 313–327.
[24] N. Agatz, A.M. Campbell, M. Fleischmann, J. van Nunen, M. Savelsbergh, Revenue management opportunities for Internet retailers, J. Revenue Pricing Manag. 12 (2013) 128–138.
[25] P. Toth, D. Vigo, Vehicle Routing: Problems, Methods, and Applications, second ed., Society for Industrial and Applied Mathematics, Philadelphia, 2014.
[26] R. Riemann, Smart Meters in Smart Homes, European Data Protection Supervisor. https://edps.europa.eu/data-protection/our-work/publications/techdispatch/techdispatch-2-smart-meters-smart-homes_en, 2019 (Accessed February 10, 2021).
[27] P. Scully, Smart Meter Market 2019: Global penetration reached 14% – North America, Europe ahead, IOT Analytics. https://iot-analytics.com/smart-meter-market-2019-global-penetration-reached-14-percent/, 2019 (Accessed February 10, 2021).
[28] S. Maldonado, R.G. González-Ramírez, F. Quijada, A. Ramírez-Nafarrate, Analytics meets port logistics: a decision support system for container stacking operations, Decis. Support. Syst. 121 (2019) 84–93.
[29] D. Pessach, G. Singer, D. Avrahami, H. Chalutz Ben-Gal, E. Shmueli, I. Ben-Gal, Employees recruitment: a prescriptive analytics approach via machine learning and mathematical programming, Decis. Support. Syst. 134 (2020) 113290.
[30] D. Veganzones, E. Séverin, An investigation of bankruptcy prediction in imbalanced datasets, Decis. Support. Syst. 112 (2018) 111–124.
[31] H. Jang, A decision support framework for robust R&D budget allocation using machine learning and optimization, Decis. Support. Syst. 121 (2019) 1–12.
[32] X. Zhang, S. Mahadevan, Ensemble machine learning models for aviation incident risk prediction, Decis. Support. Syst. 116 (2019) 48–63.

[33] F. Castanedo, A review of data fusion techniques, Sci. World J. 2013 (2013) 1–19, https://doi.org/10.1155/2013/704504.
[34] T. Meng, X. Jing, Z. Yan, W. Pedrycz, A survey on machine learning for data fusion, Inform. Fusion 57 (2020) 115–129.
[35] R. Zhang, F. Nie, X. Li, X. Wei, Feature selection with multi-view data: a survey, Inform. Fusion 50 (2019) 158–167.
[36] A. Picasso, S. Merello, Y. Ma, L. Oneto, E. Cambria, Technical analysis and sentiment embeddings for market trend prediction, Expert Syst. Appl. 135 (2019) 60–70.
[37] T. Huang, D. Bergman, R. Gopal, Predictive and prescriptive analytics for location selection of add-on retail products, Prod. Oper. Manag. 28 (2019) 1858–1877.
[38] D. Bertsimas, N. Kallus, From predictive to prescriptive analytics, Manag. Sci. 66 (2020) 1025–1044.
[39] Y. Qi, Y. Wang, X. Zheng, Z. Wu, Robust feature learning by stacked autoencoder with maximum correntropy criterion, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Florence, 2014, pp. 6716–6720.
[40] F.J. Pulgar, F. Charte, A.J. Rivera, M.J. del Jesus, Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: analysis, tips and guidelines, Inform. Fusion 54 (2020) 44–60.
[41] J. Richards, Is there a best classifier? in: L. Bruzzone (Ed.), Image and Signal Processing for Remote Sensing XI SPIE, Bruges, 2005, pp. 92–103.
[42] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[43] T. Chen, C. Guestrin, XGBoost, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2016, pp. 785–794.
[44] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Stat. 28 (2000) 337–374.
[45] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, 2016.
[46] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[47] A. Booth, E. Gerding, F. McGroarty, Performance-weighted ensembles of random forests for predicting price impact, Quantitative Financ. 15 (2015) 1823–1835.
[48] G. Seni, J.F. Elder, Ensemble Methods in Data Mining: Improving Accuracy through Combining Predictions, Morgan & Claypool, San Rafael, 2010.
[49] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (2008) 1–37.
[50] M. Kraus, S. Feuerriegel, A. Oztekin, Deep learning in business analytics and operations research: models, applications and managerial implications, Eur. J. Oper. Res. 281 (2020) 628–641.
[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, 2016, pp. 265–283.
[53] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (2009) 1263–1284.
[54] O. Loyola-González, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, M. García-Borroto, Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases, Neurocomputing. 175 (2016) 935–947.
[55] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newslett. 6 (2004) 20–29.
[56] A. More, Survey of Resampling Techniques for Improving Classification Performance in Unbalanced Datasets, ArXiv, https://arxiv.org/abs/1608.06048, 2016.
[57] P. Branco, L. Torgo, R.P. Ribeiro, A survey of predictive modeling on imbalanced domains, ACM Comput. Surv. 49 (2016) 1–50.
[58] M.W.P. Savelsbergh, Local search in routing problems with time windows, Ann. Oper. Res. 4 (1985) 285–305.
[59] A.M. Campbell, M. Savelsbergh, Efficient insertion heuristics for vehicle routing and scheduling problems, Transp. Sci. 38 (2004) 369–378.
[60] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, J.-M. Rousseau, The vehicle routing problem with time windows part I: Tabu search, INFORMS J. Comput. 8 (1996) 158–164.
[61] N. Japkowicz, M. Shah, Evaluating Learning Algorithms, Cambridge University Press, Cambridge, 2011.
[62] R.W. Sinnott, Virtues of the Haversine, S&T. 68 (1984) 158.

Shanthan Kandula is a doctoral candidate of Information Systems at the Indian Institute of Management Ahmedabad. He holds a bachelor of technology degree in Electrical & Electronics Engineering. His research interests are in e-commerce, business analytics, classical machine learning, natural language processing, and deep reinforcement learning. He also has IS practitioner experience working at banking firms.

Srikumar Krishnamoorthy is currently a faculty in the Information Systems Area at the Indian Institute of Management Ahmedabad, India. His key research interests include data mining, text mining, social media analytics, and personalization in e-commerce. He received his doctorate in IT and Systems Management from the Indian Institute of Management Lucknow, India.

Debjit Roy is a professor in production and quantitative methods at the Indian Institute of Management in Ahmedabad, India. His research interests are to estimate the performance of logistical and service systems, such as container terminals, automated distribution centers, vehicle rental, trucking, and restaurant systems. He holds a Ph.D. in Industrial Engineering (with a major in Decision Sciences/Operations Research and a minor in Computer Science) and an MS in Manufacturing Systems Engineering from the University of Wisconsin-Madison.