Prescriptive Learning for Air-Cargo Revenue Management

Stefano Giovanni Rizzo*, Yixian Chen†, Linsey Pang†, Ji Lucas*, Zoi Kaoudi‡, Jorge Quiane‡, Sanjay Chawla*
*Qatar Computing Research Institute, Doha, Qatar
{strizzo, jlucas, schawla}@hbku.edu.qa.com
†Walmart Labs, Sunnyvale, CA, USA
{yixian.chen, linsey.pang}@walmartlabs.com
‡TU Berlin, Berlin, Germany
{zoi.kaoudi, jorge.quiane}@tu-berlin.de

***Abstract*—We propose RL-Cargo, a revenue management approach for air-cargo that combines machine learning prediction with decision-making using deep reinforcement learning. This approach addresses a problem that is unique to the air-cargo business, namely the wide discrepancy between the quantity (weight or volume) that a shipper will book and the actual amount received at departure time by the airline. The discrepancy results in sub-optimal and inefficient behavior by both the shipper and the airline resulting in an overall loss of potential revenue for the airline. A DQN method using uncertainty bounds from prediction is proposed for decision making under a prescriptive learning framework. Parts of RL-Cargo have been deployed in the production environment of a large commercial airline company. We have validated the benefits of RL-Cargo using a real dataset. More specifically, we have carried out simulations seeded with real data to compare classical Dynamic Programming and Deep Reinforcement Learning techniques on offloading costs and revenue generation. Our results suggest that prescriptive learning which combines prediction with decision-making provides a principled approach for managing the air cargo revenue ecosystem. Furthermore, the proposed approach can be abstracted to many other application domains where decision making needs to be carried out in face of both data and behavioral uncertainty.**

I. Introduction

The revenue of commercial airlines is primarily derived from sales of passenger tickets and cargo (freight) shipments. While most modern airlines have implemented sophisticated data-driven passenger revenue management systems, for cargo the situation is different. The air-cargo ecosystem is complex and involves several players including shippers, freight forwarders, airlines and end-customers. Overall, there are three fundamental differences between passenger and cargo revenue management [1] [2] [3]:

(1) In the case of cargo revenue, the unit of sale is highly dynamic due to substantial variability in both volume and weight of cargo shipments. It is different from the static unit of sale, an airline seat, in the case of passenger revenue.

(2) In the air-cargo management ecosystem, there is no penalty of overbooking for customers. A large chunk of air cargo capacity is pre-booked by freight forwarders who tend to overbook and release capacity closer to the date of departure. Overbooking often leads to offloading, which has a cost in terms of storage and rerouting for the airline company.

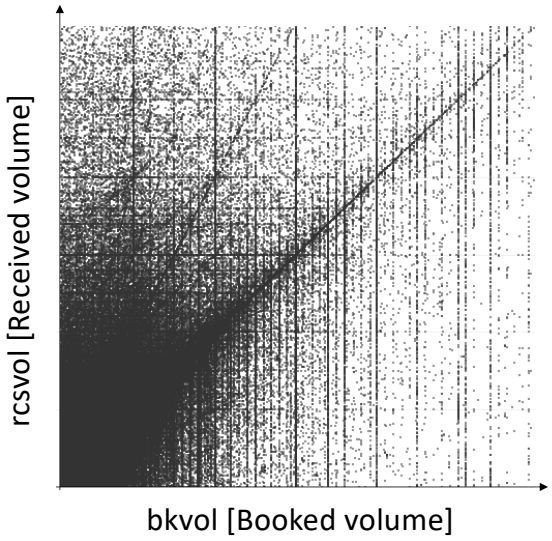

Fig. 1. The x-axis is booked volume (bkvol). The y-axis is received volume (rcsvol). In an ideal scenario, all bookings should fall in the $rcsvol = bkvol$ diagonal line. Instead, received volume widely differs from the booked volume, representing a challenge for accept/reject decision-making. For confidentiality reasons, the axis range is obfuscated.

(3) It is well known that cargo capacity is often volume-constrained, i.e., the aircraft will reach volume capacity before it reaches weight capacity. However, this makes revenue management even harder because the volume measurements are less accurate than the weight ones. Figure 1 illustrates this fact further that there exists a large discrepancy between received and booked volume.

Due to these major differences, air-cargo business requires not only to accurately predict the quantity (weight and volume) of an item that will be tendered but also to make decisions on whether to accept or reject a booking for a certain flight. This will enable the airline to greatly increase the efficiency of capacity utilization. However, prediction and decision-making in an air-cargo setting is non-trivial because of three main reasons. First, the quality of cargo booking data varies among shippers. Shippers often send information using text messages, emails, spreadsheets, voice calls, or even intermediate their

bookings through freight forwarders. Second, there are no good features for predicting the final quantity that will be received. Employees working in the cargo revenue teams use their intuition to decide whether to accept a shipment for a flight or to reroute through another flight. Third, offloading and rerouting cargo incur high costs in terms of storage and rerouting, which somehow constraints the decision-making.

We present the RL-Cargo system that deals with the above challenges. Although the use of formal decision-making is well established in the revenue management community [4], RL-Cargo is the first work that puts all pieces together to provide a complete pipeline for the air-cargo revenue management problem: given an incoming booking it (i) identifies if there might exist a substantial difference between the booked volume and the one that might be tendered, (ii) predicts the volume that will be tendered, and (iii) considers such a volume prediction to make an acceptance/rejection decision. A version of RL-cargo has been tested in production in a large airline company[1]. In particular, after surveying related work in Section II, we make the following contributions in Section III:

(1) We model air-cargo revenue management as a prediction-driven sequential and stochastic optimization problem, where the true state (total volume) is realized at departure time. We formulated it as a Markov Decision Process and then solved by using Dynamic Programming and Deep Reinforcement Learning to find the optimal policy to maximizes the revenue for each cargo flight. This model tightly integrates the above prediction model in order to make more reliable decisions. (Section III-B, Section III-C and Section III-D)

(2) We propose an uncertainty-aware Q-value function for the DQN method, to take into account the uncertainty bounds of the prediction, resulting in a safer decision making and a substantial reduction in offloading costs. (Section III-D)

(3) We evaluate the RL-Cargo using real data taken from the airline company. We demonstrate that our prediction-based decision-making technique helps the airline company to increase the total revenue and decrease the offloading cost. (Section IV)

We conclude this paper with a discussion, some lessons learned from the project, and future work in Section V.

II. Related Work

To the best of our knowledge the proposed solution is the first known cargo revenue management approach which combines prediction and decision making optimization. We overview related work in these subareas.

The discipline of Revenue Management (RM) is an advanced and well developed topic within the Operations Research (OR) community. It has roots in the airline industry and has now expanded in other areas, including hotels and tourism [4]. The initial focus in the airline industry was primarily on passenger RM, in particular how to price passenger seats in order to maximize revenue [5]. A strand of relevant work that has appeared in the data mining literature is the problem of determining overbooking rate, forecasting the number of no-shows per flight, i.e., the percentage of bookings that were made but did not show up by departure time [6], [7]. No-shows are a common problem in air-cargo too and several works have proposed solutions [8], [9].

While RM for passenger seats is now a well developed area, the same is not true for cargo management. The first research overview of issues surrounding air cargo revenue management were introduced in [10]. Since then the research literature has seen a steady growth with works including [1], [11]. Our work closely follows the decision making paradigm introduced by [3] for modeling both volume and weight aspects of bookings in air-cargo RM. The main innovation of our approach is that we integrate machine learning prediction into the decision making process using deep reinforcement learning to address the real-world cargo revenue management.

III. Proposed System: RL-Cargo

The main components of RL-Cargo are shown in Figure 2: (i) Cargo Predictor, and (ii) Decision-Maker. RL-Cargo system has offline and online modes. In online mode, given an incoming booking, the Cargo Predictor gets as input the booking from the RM system, extracts the relevant features from the booking, and predicts the volume expected to be tendered by the customer, together with a prediction interval to model the uncertainty of the decision. Then, the Decision-Maker gets the incoming booking with the predicted volume that will be tendered and takes a decision whether to accept or not the booking via our DQN module. The offline mode is composed of the process of data cleaning to generated a training seet, training of GBM model for volume prediction and DQN training in our simulated environment. We detail each of these steps in the following three sections.

A. Predicting Cargo Volume

Given an incoming booking, RL-Cargo proceeds to predict $rcsvol$ (i.e., the received volume by departure time) for the given booking, as shown in Figure 2. To do so we need to offline build a model using historical data as illustrated in the middle-bottom of Figure 2. We thus need to (i) decide on the features and (ii) decide on the algorithm to use. Predicting $rcsvol$ is quite challenging as the $bkvol$ is usually quite different from $rcsvol$. Figure 4 illustrates this difficulty. The vertical lines provide a clear indication that we need other features besides $bkvol$ to have any chance of accurately predicting $rcsvol$.

We thus experimented with extracting different feature combinations until we settled on a set that provides a good compromise between model complexity and accuracy. More formally, given a sample of bookings, we formed a feature set $\mathbf{X}$ and mapped each booking i as an element $\mathbf{x}_i \in \mathbf{X}$ and the $rcsvol$ as $y_i \in \mathbf{R}^+$. The prediction task then becomes a regression problem, where we have to learn a function

$$f_\theta : \mathbf{X} \rightarrow \mathbf{R}^+$$

[1]We omit the company name for confidentiality reasons.

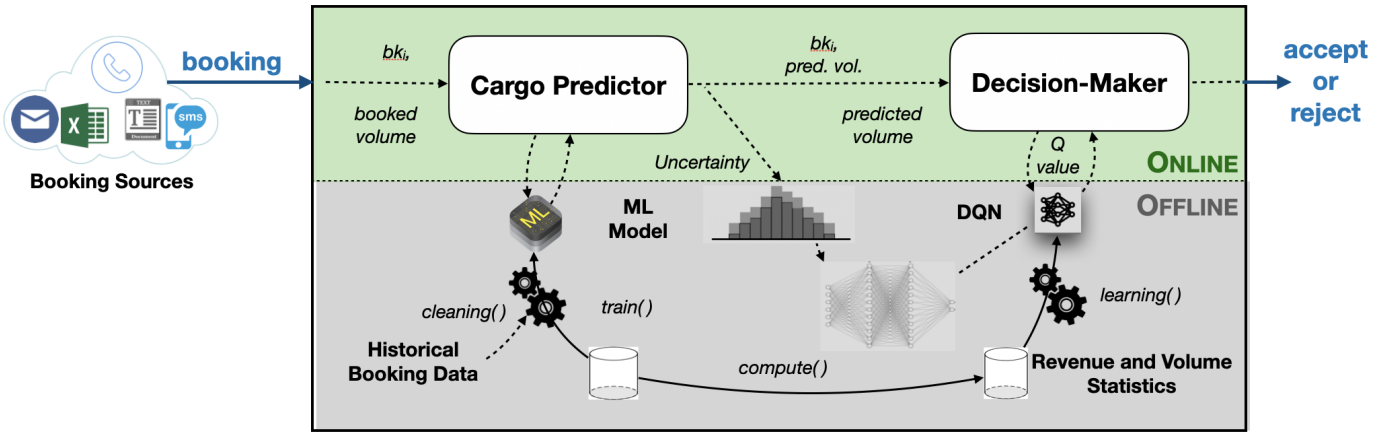

Fig. 2. The RL-Cargo DQN system mainly consists of two components: a prediction component to estimate $rcsvol$ and a decision-making DNQ engine to decide which bookings to accept in order to maximize expected revenue.

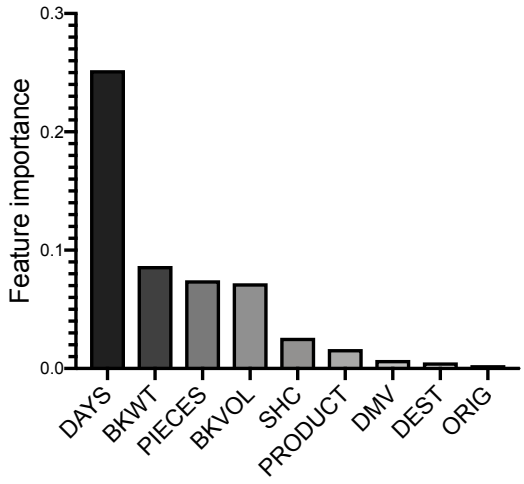

Fig. 3. Features importance. For categorical features, such as product, the value of the category with the maximum importance is shown.

Figure 3 shows the set of features we use, which were the most important ones, for predicting $rcsvol$:

- **Days until departure (DAYS):** The most important feature by far is the number of days between the booking time and the departure time. Bookings closer to the departure time tend to be more accurate. In fact, bookings that are time-stamped several days before departure day tend to show a clear pattern of overbooking from the customer side (and hence of using DMV, the flag showing the disguised missing values in data). It is natural for shippers to overbook as in the air-cargo business there is no penalty for that.
- **Booked weight (BKWT):** Contrary to $bkvol$, which usually tends to be a DMV, $bkwt$ is a valuable information. This is because shipping agents have a much more accurate information of $bkwt$ as they have access to high quality weighing machines. Indeed, instruments for accurately measuring volume are not that widespread [12], [13]. Thus, being easier to measure, $bkwt$ is on average more precise.
- **Number of pieces (PIECES):** A shipment may consist of a number of equal units. Diagonal lines in Figure 4 suggest that bookings frequently differ from the tendered shipments in the number of pieces rather than in their volume. Thus, knowing the number of pieces is useful in predicting possible outcomes at receiving time. For example, if two pieces where booked for a $bkvol$ of $12m^3$, with volume for each piece of $6m^3$, it is unlikely that a single piece will be split and the $rcsvol$ will become $4m^3$. It is in fact much more likely that it may become $6m^3$, $18m^3$, or $24m^3$.
- **Booked volume (BKVOL):** We observed that, despite of DMVs, the booked volume, $bkvol$, is still an important feature for predicting $rcsvol$. This is because when $bkvol$ is not a DMV, it tends to be precise.
- **Shipment code (SHC):** This is a set of codes to instruct how the shipment must be handled, e.g., live animals or perishables. This feature ended up being important as it specifies over (or complements the) the product type explained here below. We encode the shipment code feature as a binary vector with one element for each shipment code (one-hot encoding).
- **Product type (PRODUCT):** We observed that the patterns in $rcsvol$ vary with different product type. In theory product type should be a highly informative feature, but we observed that the distribution of product types is skewed.
- **DMV Flag (DMV):** An important observation we made is that customers often send arbitrary but fixed values as proxies for NaN. In the data cleaning literature, these proxy values are often called Disguised Missing Values (DMVs) [14], [15] Because DMVs are frequent and must be dealt within a production environment, we decided not to remove DMV data from the training set. At the same time, giving their negative impact in the prediction, it is

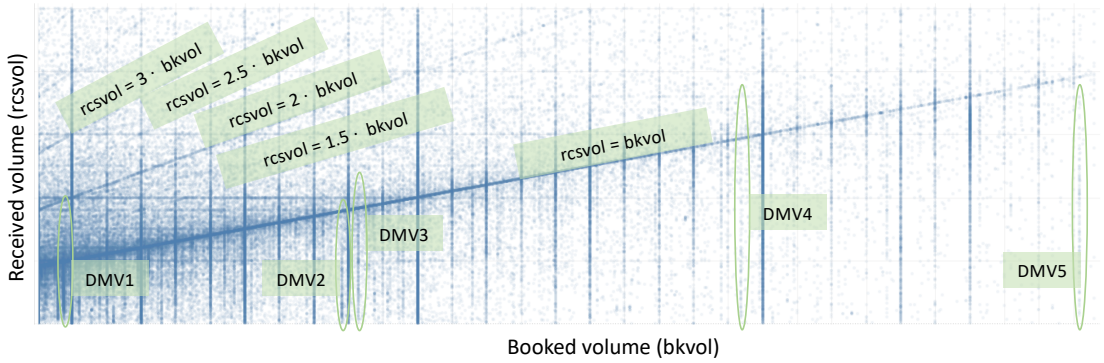

Fig. 4. Values of bkvol and related rcsvol in shipments from historical data for a given range of bkvol. Circled in green are vertical patterns that are evidently disguised missing values.

important to know if a booking has a DMV for $bkvol$. For this reason, we provide a flag, which is obtained by the DMV Detector based on historical data. Some examples of DMVs are shown in Figure 4.

- **Destination (DEST):** We also consider the destination as a feature, even if the destination alone is a weak predictor for $rcsvol$. This is because in conjunction with product type it becomes possible to elicit subtypes within products and thus reduce the variance.
- **Origin (ORIG):** The origin airport of the shipment. This feature allows capturing the average behavior of booking agents from each location.

We also experimented with using both random forests (RFs) and gradient boosting machines (GBMs) [16] for building the model. GBMs are ensemble methods and are known to perform well "out of the box". They can also easily handle a mixture of datatypes including numeric and categorical data. Recall that for a prediction problem the error can be decomposed into a sum of bias and variance [17]. On the one hand, RFs reduce the error by reducing variance as they combine independently generated deep trees on bootstrapped samples. GBMs, on the other hand, reduce the bias by building shallow trees in a sequential manner, where each subsequent tree is trained by using the dependent variable as the residuals of the previous one. In our case, even though we trained the model to make predictions at the booking level, we were primarily interested in making flight level predictions, which are an aggregation of booking level predictions. Therefore, while GBM predictions fluctuate more and individual predictions are further from the actual value, the differences cancel each other out at the flight level, that is, the aggregation of bookings at the flight level will automatically result in variance reduction. This has been confirmed by evaluating both models for booking level and flight level prediction. On the booking level, the variance in GBM predictions is more than 5 times higher than RF predictions. However, at the flight level, the mean absolute error of RF is 87.1% higher than the GBM error. For these reasons, in our production deployment we have used GBMs.

B. Model of the environment

Once the volume is predicted from $bkvol$ and the described booking features, the Decision-Maker creates an acceptance/rejection suggestion for the given booking. Before digging in how it does so, let us first state what the problem of decision making in the context of airline cargo booking is. For any flight, capacity is a perishable quantity, i.e., once the flight takes-off the capacity is lost. Therefore, an airline wants to accept bookings that will maximize revenue. The problem can be seen as a generalization of the classic Knapsack problem with two caveats: (i) cargo bookings appear over time and the exact volume (weight) of the shipment becomes available only at departure time. We, thus, model this problem as a stochastic dynamic program [3], [18].

We thus model this problem as a Markov Decision Process (MDP) [19].A Markov Decision Process is generally described by a set of all the states s referred to as state space S ($s \in S$), a set of all actions a given by the action space A ($a \in A$), a reward function $R(s, a)$ which depends on the current state and action, and a transition probability $T(s, a, s')$ that action a in state s will result in state s'. At each time t, a decision maker observes the state of the system. As a result of choosing an action in a state following the Markovian assumption, the system probabilistically evolves into a new state and the decision maker receives a reward r. A discount factor γ is generally included so that immediate rewards are valued more than rewards that could be obtained in the future. The sequential decision making problem is to find an optimal policy π^* to maximize the expected reward sequence.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[\sum_{t=0}^{T} R(s_t, a_t)|\pi]$$

Our MDP formulation is as follows:

1) State Space
   The state space (S) vector contains the information generated during the booking process regarding acceptance/rejection of cargo. It includes the volume of items that have been assigned to a flight for the i^{th} product type (x_i), the time remaining for the booking process

to departure (t), the product type of the latest item (i) assigned booking.

$$S = \{i, bkvol, x_1, x_2, ..., x_n, t\}$$

We can illustrate the state by the following example. At each time step, only one shipment booking can arrive, t days prior to departure, of booked volume $bkvol$ and product type i. Additionally, the freight shows that the volume booked in the product type 1,2,..,n are $x_1, x_2, ..x_n$ respectively.

2) Action Space
At every time step, there are one of the two decisions which are accept (a_1) or reject (a_0), that can be made. The action space (A) is given by:

$$A = \{a_0, a_1\}$$

3) Reward Functions
The reward for accepting a booking is computed using a revenue function $R(vol_i)$ based on the volume and the product type i of the booking. We follow a synthetic pricing table found in [3] to implement the revenue function. The received volume $rcsvol$ is not known before departure time, instead a proxy value for the real reward can be computed using the booked volume $R(bkvol_i)$, the average volume for type i product $R(\bar{v}_i)$, or the volume predicted using a parameterized function f_θ on the booked volume, $R(f_\theta(bkvol_i))$.
The true reward R^{true} is not observable until departure time and it is computed at each step as

$$R^{true} = \begin{cases} \text{R(rcsvol}_i\text{)}, & \text{if action} = a_1,\ t > 0 \\ \text{- h}_v(\sum_i x_i - k_v), & \text{if } \sum_i x_i - k_v > 0, \\ & t = 0, \\ 0 & \text{otherwise} \end{cases}$$

At the time of departure, if the received volume is over the maximum capacity (k_v), then the cost of off-loading occurs, which is proportional (h_v) to the total received volume minus the capacity.

4) Model Dynamics
The state space updates by the occurrence of any one of the following events: 1) new booking request, 2) flight departure. The state is reset when the terminal conditions meet (flight departure or current booking volumes reaches to maximum capacity). Actions need to be taken only when there is a booking request. In one time step there is only one item request submitted. The agent is said to be in a decision-making state at that instance. When a booking request is accepted, the volume state for the corresponding product type gets incremented by its predicted volume. When it is rejected, the volume state remains unchanged.

Defining the MDP of the model is required to address decision-making optimization problems solved via dynamic programming and reinforcement learning.

C. Dynamic programming

We start by defining a state vector of the accepted bookings, $\mathbf{x} = (x_1, \ldots, x_m)$, that can be obtained from S. Each x_i is the number of items of type i assigned to a flight. A type is a pre-defined category, like fresh food or pharma. The state $\mathbf{x}$ evolves with time t. We define the value function $VF(x, t)$ as the expected revenue from the flight given that at time t the flight is in state $\mathbf{x}$. We label departure day as time $t = 0$ and the booking horizon extends up to time $t = T$. Thus, time flows backwards. We model a single flight whose volume capacity k_v[2] is fixed and known. We also discretize time and in each time bin t the probability of an item i being received for a booking is $p_{i,t}$. We assume that at each time step only one shipment can arrive for booking. We define $p_{0,t}$ as $1 - \sum_{i=1}^{m} p(i, t)$ as the probability that no booking will show up in time period t. In practice, when an agent books an item of type i, it is accompanied by a booked volume $bkvol_i$. When the item finally arrives for shipment the received volume is $rcsvol_i$. The revenue received from the item i is $R(rcsvol_i)$, where $R()$ is typically an increasing and concave function of volume. Recall that during booking time the airline only knows $bkvol_i$ and not $rcsvol_i$. Thus, it is common to make a decision about whether to accept or reject a booking based on the average volume of type i, $\bar{v}_i$.

We can now define the value function $VF(\mathbf{x}, t)$ as a recursive function (Bellman's Equation) in order to maximize the overall expected revenue [3]:

$$VF(\mathbf{x}, t) = \sum_{i=1}^{m} p_{i,t} \max\{R(\bar{v}_i) + VF(\mathbf{x} + \mathbf{e_i}, t-1), VF(\mathbf{x}, t-1)\} + p_{0,t} VF((\mathbf{x}, t-1)), \quad t = 1, 2, \ldots, T$$

$$VF(\mathbf{x}, 0) = -h_v[\sum_{i=1}^{m} x_i \bar{v}_i - k_v]^+$$

where $[a]^+ = \max\{a, 0\}$. We now explain the above recursive equation. When the state is $\mathbf{x}$ at a given time t then $VF(\mathbf{x}, t)$ is the expected revenue over the full time horizon of the booking. At time step t, the probability of a shipment of type i arriving is $p_{i,t}$. If the booking is accepted then the state will transition to $\mathbf{x} + \mathbf{e_i}$, where $\mathbf{e}_i$ is the one-hot binary vector with a 1 at the i-th location. By accepting the booking, the expected revenue will be $R(\bar{v}_i)$. However, the booking of item i will only be accepted if the revenue $R(\bar{v}_i) + VF(\mathbf{x} + \mathbf{e_i}, t-1)$ is greater than not accepting the booking and transitioning one step towards departure while staying in the same state, i.e., $VF(\mathbf{x}, t-1)$. At time $t = 0$ and in state $\mathbf{x}$, the $VF(\mathbf{x}, 0)$ captures the cost of off-loading, which is proportional (h_v) to the total expected volume $\sum_i x_i \bar{v}_i$ minus the capacity k_v. For

[2]For non-cargo flights k_v varies depending upon passenger load.

	t=0	t=1	t=2	t=3	t=4
x=0	0.0	1.2	2.4	3.1	3.6
x=1	0.0	1.2	1.8	2.2	0.0
x=2	0.0	0.4	0.8	0.0	0.0
x=3	-1.0	-0.6	0.0	0.0	0.0
x=4	-2.0	0.0	0.0	0.0	0.0

Fig. 5. The Value Function (VF) corresponding to Example 1. Each cell corresponds to the revenue that will be accrued starting from state x at time t. Note the cell values are computed using backward induction as time is going backwards.

example, if the expected volume is 100 units and the capacity k_v is 50, then the off-loading cost is $-50h_v$.

Having defined the value function $VF(\mathbf{x}, t)$, the decision rule (D1V) at each time step t, which determines whether to accept or reject an incoming shipment of type i is given as:

$$D1V: \quad R(\bar{v}_i) + VF(\mathbf{x} + \mathbf{e_i}, t-1) > VF(\mathbf{x}, t-1)$$

However, we can integrate prediction in the decision-making by modifying the decision rule. For example, suppose our predictive function is f_θ (as defined in Section 3.2[3]), i.e., given a booked volume $bkvol_i$ of type i, $f_\theta(bkvol_i)$ is the predicted received volume ($\hat{rcsvol_i}$). We, then, have a new decision rule (D2V) as

$$\begin{aligned} D2V: \quad & R(f_\theta(bkvol_i)) \\ & + VF(\mathbf{x} + \mathbf{e_i}, t-1) > VF(\mathbf{x}, t-1) \end{aligned}$$

The Curse of Dimensionality: It is worth noting that the construction of $VF(\mathbf{x}, t)$ suffers from the well-known curse of dimensionality of dynamic programming [18], [20]. For example, suppose there are m items and the number of time periods is T. Then, the size of the state space is exponential in m[4]. An approximate solution to escape the exponential blow-up is to use aggregate $x = \sum_i x_i \bar{v}_i$. This makes the state space one-dimensional scalar-valued, instead of vector-valued, of maximum size. This state space is bounded by $M \times T$, where M is the maximum possible volume booked for any type. The construction of $VF(x, t)$ becomes considerably simplified and the decision rule D2V then becomes D2S:

$$\begin{aligned} D2S: \quad & R(f_\theta(bkvol_i)) \\ & + VF(x + f_\theta(bkvol_i), t-1) > VF(x, t-1) \end{aligned}$$

In the same way, we define D1S as simplified state version of D1V, where only booked volume is observed without access to prediction.

[3]We have overloaded the f_θ signature to emphasize the role of $bkvol$

[4]It is $S(T, m)$, Stirling number of second kind

Illustrative Example 1: To illustrate how dynamic programming is used to form the VF function, we will work through a simple scenario shown in Table I.

TABLE I
SAMPLE CARGO CHARACTERISTICS

		type1	type2
1	volume	1	1
3	revenue (ρ)	1	2
4	prob. arrival in t	0.4	0.4
5	prob. no booking in t	0.2	
6	max capacity (k_v)	2	

We assume there are two shipment types: type1 and type2. Both types can arrive for a booking with a probability of 0.4 in any time step and the probability that no shipment will arrive for any booking is 0.2. The revenue for type1 is 1 and for type2 is 2, while the volume for both types is fixed at 1 unit. Recall that time is labeled in a reverse order, i.e., departure time is 0 and booking horizon extends up to time $t = 4$. To compute the value function VF, we proceed backwards for each state $\mathbf{x}$. Now the state is a two-dimensional vector $\mathbf{x} = (x_1, x_2)$, where x_1 and x_2 are the number of bookings of type1 and type2 respectively. However, we collapse $\mathbf{x}$ into $x = x_1\bar{v_1} + x_2\bar{v_2} = x_1 + x_2$ as we have assume that the volume booked has a value of 1. The different values of x are shown as rows in Figure 5. We first have to populate the first column of Table I. For example, $VF(0, 0) = -\max(0 - k_v, 0) = 0$ as $k_v = 2$ and $VF(3, 0) = -\max(3 - 2, 0) = -1$. As an example, we compute $VF(1, 2)$.

$$\begin{aligned} VF(1,2) =& 0.4 \max(1 + VF(2,1), VF(1,1)) \\ & + 0.4 * \max(2 + VF(2,1), VF(1,1) \\ & + 0.2 * VF(1,1) \\ =& 0.4 * (1 + 0.4) + 0.4 * (2 + 0.4) + 0.2 * 1.2 \\ =& 1.76 \approx 1.8 \end{aligned}$$

D. Deep Q-Learning

Dynamic programming methods are well-developed mathematically, but require a complete and accurate model of the environment [21]. In our case, the cargo prediction results have uncertainty, which can lead to an undesirable Dynamic Programming performance. We will address the optimal decision-making under uncertainty using Reinforcement Learning. Q-learning [22] is one of the most popular reinforcement learning algorithms to solve MDP using action-value function. Action-value function returns the expected true value (Q-value) of an action in a state under a given a policy. A true value of an action is the mean reward when that action is selected. The optimal action at each state is the one that maximizes the state-action value.

$$\begin{aligned} Q_{t+1}(s_t, a_t) =& Q_t(s_t, a_t) + \alpha[R_t + \gamma \max_{a_t} Q_t(s_{t+1}, a_{t+1}) \\ & - Q_t(s_t, a_t)] \end{aligned}$$

In the context of Q-learning, an effective approach to address the curse of dimensionality of exponentially increasing state space is to use the *function approximation* and *representation learning* features of deep learning. The agent does not need to visit all states, and the state-action values that have not been encountered would be generalized from neural networks. Here we will use Deep Q-Networks (DQN) [23] to model the problem. For n-dimensional state space and an action space containing 2 actions, the neural network is a function from $\mathbb{R}^n$ to $\mathbb{R}^2$. DQN combines Q-learning with two networks: prediction network and target network. The target network with parameters θ^-, is the same as the online network except that its parameters are copied every τ steps from online network, so that then $\theta_t^- = \theta_t$ and kept fixed on all other steps. Another important ingredient in DQN is experience replay. The experience replay observed transitions that are stored for some time and sampled uniformly from this memory bank to update the network. Both the target network and the experience replay dramatically improve the performance and stability of the algorithm. In order to ensure that model converges to the optimal value, some amount of exploration is required depending upon the known information of the environment, therefore an ϵ-greedy policy is implemented with an exponential decay of the ϵ probability through the training episodes.

Uncertainty: Uncertainty intervals from prediction models can be used to further improve the safety of the decision making, with the goal of reducing offloading cost and therefore increase the revenue. We integrate the uncertainty of the prediction, in the form of prediction intervals $[b_l, b_u]$, in the following way. Given the current state using predicted volume $S = \{i, f_\theta(bkvol), x_1, x_2, ..., x_n, t\}$, we define S_l and S_u

$$S_l = \{i, b_l, x_1, x_2, ..., x_n, t\}$$
$$S_u = \{i, b_u, x_1, x_2, ..., x_n, t\},$$

then, we define the uncertainty-aware Q function as

$$Q_U(S, a) = \frac{1}{2}\left(Q(S_l, a) + Q(S_u, a)\right).$$

The uncertainty-aware $Q_U(S, a)$ is particularly useful when the value of the upper or the lower bound state of the prediction significantly deviates from the value of the predicted state. Consider as an example of the case when accepting a volume of b_u would incur in very high offloading costs, while accepting a volume of $f_\theta(bkvol)$ or b_u would lead to a moderate revenue, that is $Q(S_u, a_1) \ll Q(S, a_1) < Q(S_l, a_1)$. In this case $Q_U(S, a_1) < Q(S, a_1)$ thus the agent will less likely accept the booking and opt for the safer action.

IV. Results

We first evaluate our proposed RL-Cargo system in terms of revenue and costs. We then show an in-depth analysis of our techniques. Note that the prediction module of RL-Cargo has been deployed in a large international airline company and the results reported are from the production environment.

A. Dataset

We obtained a real dataset spanning two years (June 2016-August 2018) of booking records from the cargo IT team of the airline company. Each booking record consists of several attributes including, booking date, origin, destination, agent, booking volume ($bkvol$), product type, received date, departure datetime, and received vol ($rcsvol$). We use this dataset to detect the DMVs and build the ML model for predicting $rcsvol$ using all other attributes. As we do not have real information on the revenue and offload costs, we create simulated data as proposed in [3] to evaluate our decision making approach. To create the simulations from the real dataset, we compute the probabilities of the product types from the real dataset: For each type i, we compute

$$p_i = \frac{\text{\# bookings with product } i}{\text{\# total bookings}}$$

The related probabilities of the ten most frequent product types are shown in the top Table II. We observe that the product type frequencies are skewed, with the most frequent product type with a probability of 0.856. Then, we split the booking horizon into 60 equal time steps and we compute the probability of a booking arriving at time step t as

$$p_t = \frac{\text{\# shipments at time } t}{\text{\# shipments in dataset}}$$

For the sake of simplicity and to replicate the same formulation as in [3], we compute a single probability for 6 different intervals of time steps, resulting in 10 time steps per interval. For each interval we take the average of the 10 single time steps that belong to this interval. The results are shown in the bottom of Table II. Given p_i the probability of an incoming type i at any time, and p_t the probability of getting any type of booking at time t, the probability of getting a booking of product type i at time t is $p_{i,t} = p_i p_t$.

B. Predictive evaluation

We first evaluate our prediction module. For our evaluation, we use 3-fold cross-validation on the full real dataset of two years cargo bookings. We make a prediction on each single booking and we evaluate the aggregated flight leg predicted volume vs. the flight-leg received volume. For this reason, we implemented cross-validation so that all the bookings from the same flight leg are kept in the same split. Based on grid-search results, we set the XGBoost regressor with 0.9 subsample ratio of columns for each split, 300 estimators, a maximum tree depth of 20 and a learning rate of 0.05. All other parameters are set as default.

We use the mean relative absolute error on our predictive model: $e = \frac{1}{N}\sum_1^N \frac{|rcsvol_i - f_\theta(bkvol_i)|}{rcsvol_i}$. The average error on the entire historical data is 7.8%. Figure 6 shows that the prediction error is under 5% in almost half of the flights, while it is under 10% for the 74.8% of the flights. By using the predictive model instead of the actual booked volume values we have a greater number of flights that have a small error. It

TABLE II

THE TOP TABLE SHOWS THE PROBABILITIES (p_i) OF MAKING A BOOKING FOR A PRODUCT TYPE i. THE BOTTOM TABLE SHOWS THE PROBABILITIES (p_t) OF A BOOKING ARRIVING IN A TIME PERIOD t.

Product type	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type10
p_i	0.856	0.042	0.036	0.035	0.012	0.007	0.005	0.003	0.002	0.002

Time period	1-10	11-20	21-30	31-40	41-50	51-60
p_t	0.05	0.03	0.009	0.004	0.003	0.005

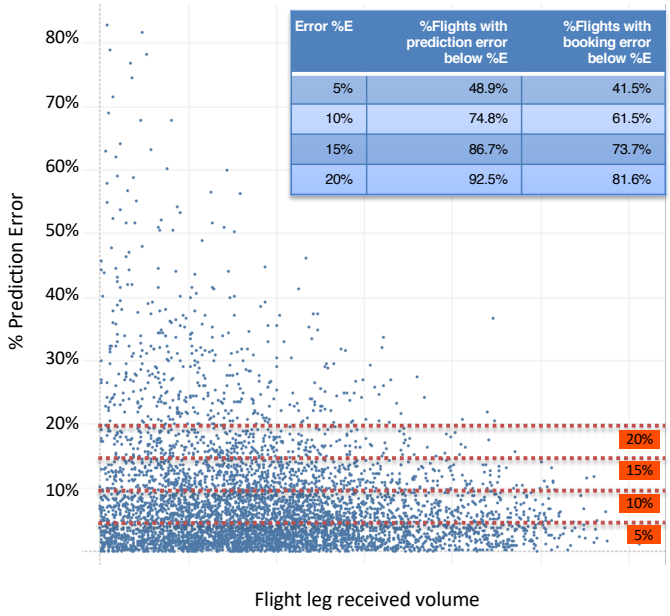

Fig. 6. Prediction error for flight legs in test splits. Each data point is a flight leg. Below each dotted line are all the flights with an error below the respective percentage.

is also noteworthy that our prediction error is lower for higher capacity flights, where it has the biggest impact.

TABLE III

AVERAGE ERROR DECREASE FOR EACH PRODUCT TYPE, AT BOOKING LEVEL, IN PRODUCTION ENVIRONMENT.

Product type	% of booking	% error decrease from bkvol
Type1	73.3%	-33.4%
Type2	10.1%	-38.3%
Type3	3.7%	-61.9%
Type4	3.5%	-66.9%
Type5	3.1%	-34.1%
Type6	2.7%	-7.2%
Type7	1.4%	-43.9%
Type8	0.8%	-96.6%
Type9	0.7%	+25.8%
Type10	0.1%	-30.7%

Table III shows the benefits of the prediction on the shipment level for the 10 most frequent product types. Specifically, it shows the decrease in percentage of the prediction error (predicted $rcsvol$ vs. actual $rcsvol$) from the booking error (original booked volume vs. actual $rcsvol$). The predicted volume has decreased the error considerably for 9 out of the 10 product types. The increase in error for product type9 is on a very rare product type and, thus, there are not enough data to train the model. However, as it is rare it does not influence the total flight-leg predicted volume.

Impact of prediction. We evaluate the impact of using predicted volume, instead of booked volume, in terms of revenue benefits and offload costs using dynamic programming for decision making. First, we compute the $VF(x,t)$ table by using the entire dataset and taking the booking volume for each time step. Then, we consider two different test cases in order to evaluate the power of our system by combing predictive modeling with decision making:

(1) *BKD to RCS*: no prediction is made. The decision to whether accept or reject an incoming booking by applying $DS2$ is based on the reported bkvol, i.e., $f_\theta(bv_i) = bv_i$. Final offloading cost is then calculated based on the rcsvol.

(2) *PRED to RCS*: The received booking is processed for DMV identification and the resulting feature vector is used to output a prediction $f_\theta(bv_i)$ using our prediction model. The decision to accept or reject an incoming shipment using $D2S$ is based on this predicted volume.

At each time step we draw bookings from the dataset following the probabilities of Table II and apply the decision rule $D2S$. Figure 7 shows the results on eleven different flight capacities k_v and ten thousand flights each, for a total of 220 thousand flights.

The top graph of Figure 7 clearly shows that for various capacity constraints (k_v) the offloading cost is lower almost by a factor of ten and with a much lower standard deviation. This suggests that using predictions instead of booked volume ($bkvol$) not only reduces the offloading cost but adds substantial amount of certainty into the whole air cargo booking process. In the bottom graph of Figure 7, we show the final revenue, i.e., after subtracting the offloading costs, for various flight capacity constraints. We observe that the revenue increases when using a predicted booked volume, albeit slightly, indicating that the decision function selected better-value shipments during the booking time horizon. Still the standard deviation of the revenue is lower when using the predictions. Note that, the way the decision making process is designed, excess overbooking incurs negative penalty (i.e., offloading), while underbooking results to zero penalty (i.e., $VF(x,0)$ is zero when the total volume is less than the flight capacity). It is thus more beneficial to reduce the risk

TABLE IV
EVALUATION OF PRESCRIPTIVE RL ON A TESTING SET OF 1000 FLIGHTS. *Booked* MODELS HAVE ACCESS ONLY TO BOOKED VOLUME OF EACH BOOKING, WHILE *Prediction* MODELS HAVE ACCESS TO THE PREDICTED VOLUME FROM THE GBM.

Model	Observed load	True load	Observed revenue	Offloading cost	**Final revenue**
FCFS (Booked)	0.99	0.47	176.31	2.08	82.27
DQN (Booked)	1.04	0.52	183.44	2.53	91.25
FCFS (Prediction)	0.99	0.69	177.61	7.65	115.9
D2S (Prediction)	1.14	0.79	198.64	1.54	117.86
DQN (Prediction)	1.09	0.75	195.17	10.7	123.74
DQN with uncertainty (Prediction)	1.09	0.73	194.78	5.97	**124.84**

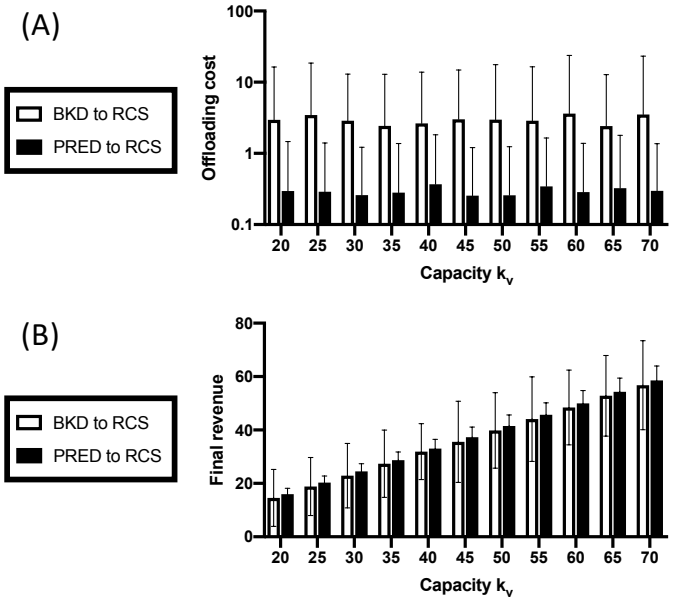

Fig. 7. Offloading cost (A) and final revenue (B) for varying levels of capacity k_v for the two dynamic programming proposed methods D1S (BKD to RCS) and D2S (PRED to RCS), showing the positive impact of using the prediction output. Note: scale of offloading cost is logarithmic.

of offloading by using a predictive model that overpredicts leading to less shipments getting accepted. This is a design choice that is driven by business objectives.

C. Prescriptive evaluation with DQN and uncertainty

We now evaluate the overall prescriptive solution using Reinforcement Learning with a DQN and uncertainty-aware Q-values. The following models are considered in the comparison:

1) FCFS (Booked): a first-come first-served (FCFS) policy using the booked volume for decision making. In the FCFS policy, every incoming booking is accepted until the capacity, estimated using $bkvol$, runs out. FCFS is a greedy strategy in the sense that it will accept immediate revenue instead of waiting for a potential booking from which more revenue can be made.
2) FCFS (Prediction): a FCFS policy using the predicted volume $f_\theta(bkvol)$ for decision making.
3) D2S (Prediction): the dynamic programming model defined in Section III, using predicted volume $f_\theta(bkvol)$ for decision making.
4) DQN (Booked): the DQN model trained and tested using the booked volume $bkvol$ as observable variable.
5) DQN (Prediction): the DQN model trained and tested using the predicted volume $f_\theta(bkvol)$ as observable variable.
6) DQN with uncertainty (Prediction): the DQN model, trained on prediction, using the proposed uncertainty-aware Q-values for decision making on the predicted volume $f_\theta(bkvol)$ and the prediction intervals $[b_l, b_u]$.

In our experiments, the architecture of the DQN is a fully connected network with 3 hidden layers of dimensions [128, 64, 16]. Training is performed in batch of 64 bookings tuples, sampled from a replay buffer of 1024 tuples. All the DQN models are trained for 800'000 steps and around 10'000 episodes of variable lengths (Figure 8). Each episode corresponds to one flight, while bookings are randomly sampled at each timestep from historical data using the probabilities in Table II. The testing set is a dataset of 1000 flights following the same design and probability distribution used to generate episodes in the training, however the flights are newly generated and cannot be found in the training set.

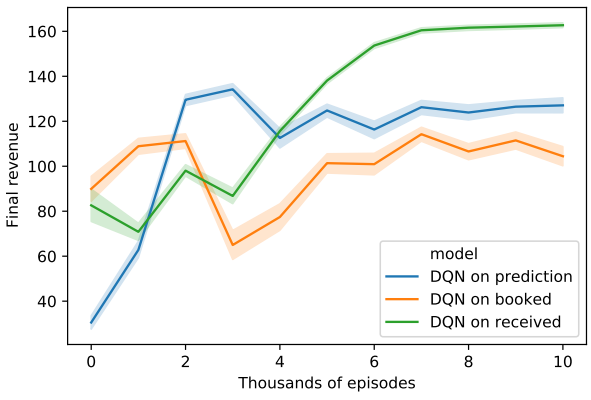

Fig. 8. DQN Performance during training: As expected, rewards using predicted volume is higher than booked and lower than received. Note, DQN agents using received volume cannot be implemented in practice.

Results. In Table IV we summarize the results on the testing set, sorted by ascending value of final revenue. As expected the policies that rely on the booked volume for decision making are the worst-performing, confirming the results of the dynamic programming experiments. However the DQN model on booked volume outperform the related FCFS with a 9% improvement on the final revenue.

Note that FCFS (Booked) and FCFS (Prediction) have an observed load of 99%. If the booking value was ideally equal to the received value, FCFS will not incur any offloading costs. However, since the two values are rarely the same, the advantage of FCFS is not realized and the experiments bear that out: the average true load is much less then observed, with 47% and 69% respectively for booked and prediction, while it also incurs in offloading costs for some of the flights.

The advantage of $D2S$ over FCFS (Prediction), despite an observed load of 114%, shows how the decision rule keeps overbooking if the revenue rate is advantageous, while rejecting the less profitable shipments once the capacity is reached. Both the proposed DQN models using the predicted volume outperform other policies despite having a lower true load than $D2S$, suggesting a better ability in picking profitable bookings based on prediction, product type and time left.

Finally, the DQN model with uncertainty on predicted volume has obtained a slightly higher reward than the simple DQN while accepting on average less true volume. Since the observed load is the same as the DQN (Prediction), while the true load is 4% less, it follows that the DQN with uncertainty tends to accept more the overpredicted bookings. Consequently at departure time the offloading cost is reduced by 44%, showing how the uncertainty-aware Q-value leads to safer actions which in turn leads to higher final revenue.

V. Discussion and Future Work

In this paper we have described the components of RL-Cargo, an intelligent and data-driven air-cargo revenue management system that combines predictive analytics and decision making under a prescriptive learning framework. RL-Cargo was developed in conjunction with a large commercial airliner over a two year period.

The project started with the stated objective of predicting received volume (rcsvol) of air-cargo bookings as most flights are volume constrained, i.e., they run out of volume space before weight capacity. However, over time, we realized that prediction per se cannot be carried out in isolation. We have to analyze the upstream sources of data and understand how the data was being generated. We also had to get a better understanding of how the outcome of the prediction task will be consumed by end users for decision making. This led us to formulate the prediction-driven revenue optimization problem.

First a dynamic programming solution was proposed to optimize the decision making based on the predicted volume. The restrictions imposed by the model-based method led us to design a Reinforcement Learning approach that could incorporate both prediction uncertainty and decision making with the final goal of optimizing the revenue.

Our general conclusion is that in order to make real and tangible impact, data science techniques have to be situated and combined with an overall objective. For future work we plan to extend the RL-Cargo system so that it can be used by shippers and freight-forwarders and not just the revenue management teams within an airline.

References

[1] A. Popescu, "Air cargo revenue and capacity management," Ph.D. dissertation, Georgia Institute of Technology, 2006.

[2] T. Boonekamp, J. Gromicho, W. Dullaert, and B. Radstaak, "Air cargo revenue management," *Unpublished masters thesis, Vrije Universiteit, Amsterdam*, 2013.

[3] K. Amaruchkul, W. L. Cooper, and D. Gupta, "Single-leg air-cargo revenue management," *Transportation science*, vol. 41, no. 4, pp. 457–469, 2007.

[4] W.-C. C. Chiang, J. Chen, and X. Xu, "An overview of research on revenue management: current issues and future research," *International Journal of Revenue Management (IJRM)*, vol. 1, no. 1, 2007.

[5] J. I. McGill and G. J. V. Ryzin, "Revenue Management: Research Overview and Prospects," *Transportation Science*, vol. 33, no. 2, pp. 233–256, 1999.

[6] R. D. Lawrence, S. J. Hong, and J. Cherrier, "Passenger-based Predictive Modeling of Airline No-show Rates," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 397–406.

[7] C. Hueglin and F. Vannotti, "Data Mining Techniques to Improve Forecast Accuracy in Airline Business," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01, 2001, pp. 438–442.

[8] Y. Lan, M. O. Ball, and I. Z. Karaesmen, "Regret in overbooking and fare-class allocation for single leg," *Manufacturing & Service Operations Management*, vol. 13, no. 2, pp. 194–208, 2011.

[9] A. Popescu, P. Keskinocak, E. Johnson, M. LaDue, and R. Kasilingam, "Estimating air-cargo overbooking based on a discrete show-up-rate distribution," *Interfaces*, vol. 36, no. 3, pp. 248–258, May 2006.

[10] R. G. Kasilingam, "Air cargo revenue management: Characteristics and complexities," *European Journal of Operational Research*, vol. 96, no. 1, pp. 36–44, 1997.

[11] S. Budiarto, H. P. Putro, P. Pradono, and G. Yudoko, "Revenue management of air cargo service in theory and practice," *IOP Conference Series: Earth and Environmental Science*, vol. 158, pp. 12–22, 2018.

[12] B. Slager and L. Kapteijins, "Implementation of cargo revenue management at klm," *Journal of Revenue Pricing Management*, vol. 3, pp. 80–90.

[13] D. Beidermand, "New freight dimensions: For shippers, finding the real cost of a shipment shouldn?t be a matter of weight and see," *Air Cargo World*, vol. 92, no. 8, pp. 34–40, 2002.

[14] R. K. Pearson, "The Problem of Disguised Missing Data," *SIGKDD Explor. Newsl.*, vol. 8, no. 1, pp. 83–92, 2006.

[15] A. A. Qahtan, A. Elmagarmid, R. Castro Fernandez, M. Ouzzani, and N. Tang, "Fahes: A robust disguised missing values detector," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2100–2109.

[16] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[17] T. Tibshirani and J. Friedman, *Elements of Statistical Learning*, 2009.

[18] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[19] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, 1957.

[20] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 2005, vol. 1, no. 3.

[21] A. G. B. Richard Sutton, *Reinforcement learning: An introduction*. MIT press, 2018.

[22] D. P. Watkins, C.J., "Q-learning," *Machine Learning*, pp. 279–292, 1992.

[23] e. a. Mnih, Volodymyr, "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*, 2013.