AI for Food Rescue

Zheyuan Ryan Shi,[1,3] Yiwen Yuan,[1⋆] Kimberly Lo,[1⋆]Ameesh Kapoor,[2]
Anthony Levin-Decanini,[2] Sean Hudson,[2] Jake Tepperman,[2] Leah Lizarondo,[2]
Fei Fang[1]

[1]Carnegie Mellon University
[2]412 Food Rescue
[3]98Connect

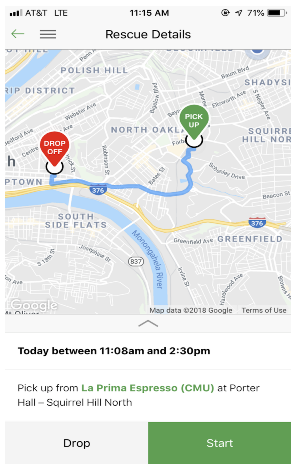
Abstract. The challenges of food waste and insecurity arise in wealthy and developing nations alike, impacting millions of livelihoods. A major force to combat food waste and insecurity, food rescue organizations match food donations to the non-profits that serve low-resource communities. However, they rely on external volunteers to pick up and deliver the food, which bring significant uncertainty to the food rescue operation. We work with 412 Food Rescue, a large food rescue organization, to predict and address this uncertainty. We make the following contributions. (1) We train a stacking model which predicts whether a rescue will be claimed with high precision and AUC. This model can help the dispatcher better plan for backup options and alleviate their uncertainty. (2) We develop a data-driven optimization algorithm to compute the optimal generic intervention and notification scheme applicable to all rescues. The recommended actions have been adopted by 412 Food Rescue and are shown to have improved the rescue efficiency. (3) We develop a rescue-specific recommender system to send push notifications to the most likely volunteers for each given rescue. We leverage a mathematical programming based approach to diversify our recommendations, and propose an online algorithm to dynamically select the volunteers to notify without the knowledge of future rescues. Our recommendation system improves the hit ratio from 44% achieved by the previous method to 73% on historical data. A randomized controlled trial of this method is scheduled to take place in the near future.

Keywords: Food security · Food waste · Machine learning · Recommender system.

1 Introduction

Food waste and food insecurity exist in many places around the world. For example, in the US, over 25% of the food is wasted, with an average American wasting about one pound of food per day [3]. Meanwhile, 12% of American households struggle to secure enough food at some point [2]. With the COVID-19 pandemic, the problem is becoming even worse [5]. Fortunately, from New

⋆ Work done while at CMU.

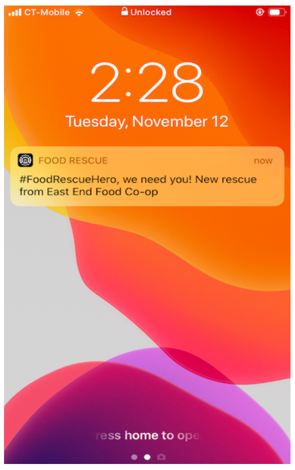


Today between 11:08am and 2:30pm

Pick up from La Prima Espresso (CMU) at Porter Hall – Squirrel Hill North

(a) Screenshot of a rescue task

(b) Donation picked up at a cafe



#FoodRescueHero, we need you! New rescue from East End Food Co-op

(c) App notification

Fig. 1. 412 Food Rescue operations

York to Colorado, from San Francisco to Sydney, food rescue (FR) platforms are fighting against food waste and insecurity in over 100 cities around the world. In the US alone, there are already over 50 cities where FRs are providing basic necessities to the communities, affecting over a million people. Their operation has proved to be effective [7]. Food rescue organizations receive edible food from restaurants and groceries (referred to as “donors”) and send it to organizations serving low-resource communities (“recipients”). These food rescue organizations are an important force to fight against food waste and food insecurity, both included in the United Nations’ Sustainable Development Goals.

Food rescue organizations serve as an intermediary between the food donors and recipient organizations. As an example, we describe how 412 Food Rescue (412FR), a large food rescue organization in Pittsburgh, US, [1] connects donors with recipients, and delivers the food with the help of volunteers. Many other FR platforms run in similar ways. Donors would call 412FR when they have food items that they want to donate. After receiving the call, the FR dispatcher matches this donation with a recipient organization. Once this matching is done, the dispatcher posts this matching on 412FR’s mobile app. Hereafter, the food rescue process becomes visible to the volunteers who have the FR’s mobile app installed on their phone (Fig. 1a). If they choose to claim a rescue on the app, the app would provide them with the detailed information instructing them where to pick up the donation and where to deliver. The volunteer then goes out to complete the rescue trip (Fig. 1b).

[1] `https://412foodrescue.org/`

Relying on volunteers saves cost for the food rescue organization, but volunteers also bring a high degree of uncertainty as to whether a rescue would be claimed or not. Occasionally, some rescue trips stay unclaimed on the mobile app for a long time. FR dispatchers want to avoid this situation as much as possible, since unclaimed rescues not only lead to immediate food waste, and also discourage the donors and recipients from participating in the program in the long run. Over the years, 412FR has mainly used two methods to get more rescues claimed. The first one is push notifications. After a rescue request is posted, the FR's smart phone app will push notification to volunteers within 5 miles (Fig. 1c). After 15 minutes, if no one has claimed the rescue, the app will push notification to all available volunteers. However, while it is helpful to engage with the volunteers, too many notifications might drive them away [4]. The second method is proactive intervention. The dispatchers monitor all the outstanding rescues. If no one has claimed the rescue by the last hour of its pickup window, the dispatcher will call the *frequent* volunteers they are familiar with to help with the rescue. As such, the dispatcher has a heavy workload.

In this chapter, we introduce the three AI-based modules that we developed to address the uncertainty issue in food rescue operations. These modules are developed jointly by AI researchers at Carnegie Mellon University and practitioners at 412FR.

First, to address the uncertainty we first need to understand the uncertainty. In Section 2.1, we train a stacking model to predict whether a rescue would be claimed. Our stacking model achieves an AUC of 0.81, serving as a reliable reference of the risk of a rescue. The model informs the dispatcher how likely a rescue is going to be claimed, thus helping the dispatcher better plan for backup options.

Second, we perform data-driven optimization to find the optimal *Intervention and Notification Scheme* (INS), i.e., when the dispatcher should intervene and seek help from regular volunteers and when and to whom the notifications should go out (Section 2.2). We follow the form of notification scheme that 412FR has been using: send the first batch of notification to volunteers within a certain distance of the pickup location of the rescue, and wait for a certain period of time before sending the second back of notification to all the available volunteers. However, we optimize the distance and waiting time. We estimate the counterfactual rescue outcomes and use a branch-and-bound method to improve computational efficiency. The resulting INS can improve over the current practice by reducing the number of notifications sent and the dispatcher interventions, while keeping the rescues' expected claim rates. Our analysis suggests to the platform some changes in their current INS, which can save the most valuable resources to food rescue: the dispatcher's attention and volunteers' interest. The changes have been implemented in the mobile app of 412FR since February 2020.

Third, as a departure from the generic notification scheme above, we develop a rescue-specific push notifications scheme (Section 2.3). By treating each rescue trip as a "user" and each volunteer as an "item", we develop a recommender system to send push notifications to the most likely volunteers for each given rescue. We then leverage a mathematical programming-based approach to diversify our recommendations. We also propose an online algorithm to dynamically select the volunteers to notify without the knowledge of future rescues. Our recommendation system improves the hit ratio from 44% achieved by the previous method to 73%. A controlled experiment for comparing the previous notification scheme and this recommender-system-based notification scheme is on the way.

2 Method

2.1 Predicting the Claim Status of Rescues

Our first task is to predict whether a rescue would be claimed. We use the database of 412FR which contains rescues from March 2018 to May 2019. The dataset records the time log of each step in the rescue: posting, claimed by volunteer, and completion, along with the ID of the volunteer who claimed the rescue. We treat a rescue as unclaimed and assign a negative label if it was never claimed or if it was claimed within the last hour of the pickup window by a selected group of volunteers who had done more than 10 rescues within the last two months. We assume the latter ones had gone through dispatcher's intervention and would not have been claimed otherwise. The dataset contains 4574 rescues with 749 negative ones among which 672 were not claimed by anyone and 77 were claimed within the last hour of the pick up window by the selected group.

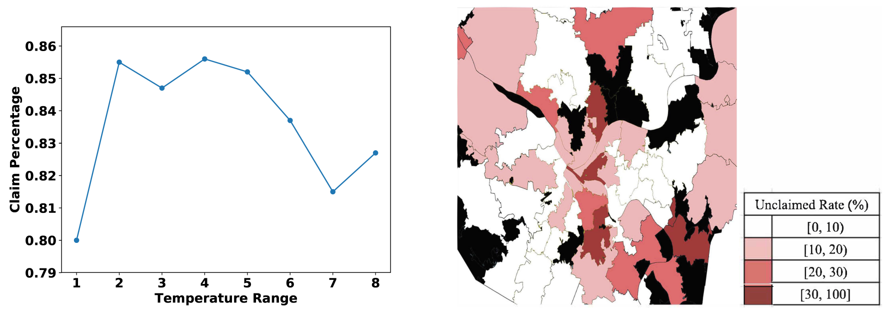
2.1.1 Feature Engineering

We use a number of features for the prediction. The first group of features are directly related to the rescue, such as the travel time and distance between the donor and the recipient generated by Google Maps Platform, the weight of the food, time of day, and which time slot the rescue belongs to.

We also used the weather information on the day of rescue from Climate Data Online[2], including the average temperature, precipitation and snowfall, as data analysis suggests that weather is correlated with the rescue outcome (Fig. 2a).

The third group of features involve the number of available volunteers near the donor and recipient's locations. Instead of using zip code, we evenly divide the area of operation of 412FR into a grid with 300 cells because the zip code districts vary a lot in size (Fig. 2b). Each volunteer could set in their app the time slots they do not want to receive any notifications, which can also be interpreted as their availability. An *active* volunteer (AV) in a grid cell for a rescue is one who had done a rescue in the cell and marked themselves as available for the rescue's pick-up time slot in the app. We use as feature the number of AVs in the donor's and recipient's cells and the number of them averaged over the cells adjacent to the donor's. The number of AVs who indicate they have vehicles

[2] `https://www.ncdc.noaa.gov/cdo-web/`



(a) Claim rate vs. temperature.

(b) Percentage of unclaimed rescues by zip code district.

Fig. 2. Data analysis results. The temperature range i represents $(10.5i - 11.5, 10.5i - 1]$°F.

is helpful as well, as those without vehicles might be more constrained in their choice of rescues.

We also tested some other features such as average household income and vehicles. However, they do not improve the performance of the model. An example of the features we use for the training the machine learning model is shown in Table 1. These two data points are for illustration purpose and are not real rescues, as per our agreement with 412FR.

2.1.2 Stacking Model

We first attempted a few baseline models including Gaussian Process (GP) and Random Forest (RF) with different parameters but got unsatisfying performance, especially with the false positives, i.e. when the rescue is unclaimed but we predict it as claimed. In the context of food rescue, we want to inform human dispatchers which rescues will be unclaimed without human intervention and need extra attention. Thus, false positives can be costly because it may lead to the ignorance of a rescue in need of intervention and the waste of donated food, while false negatives are less concerning because it only leads to unnecessary extra attention from the human dispatcher. To deal with weak learners, we use a stacking approach inspired by [8], whose structure is shown in Fig. 3. First, we split the training data into two sets, D_A and D_B. We use D_A to train various base models (Fig. 3, ①) and then we use these trained models to make predictions on D_B (Fig. 3, ②). Finally, we train a meta learner using the base models' predictions on D_B to determine the stacking model's estimate (Fig. 3, ③). In our case, we use 5 GP regressors and 1 RF classifier as the base model. The 5 GPs have different kernels and parameters for length scales. The parameters for GPs are shown in Table 2. The Random Forest Classifier has 100 estimators and the max-depth for any decision tree is 9.

Features	Rescue 1	Rescue 2
Fastest travel time of rescue	8 min	28 min
Travel distance of rescue	2.4 miles	18 miles
Weight of the food	5 lb	20 lb
Time of day	1pm	2pm
Time Slot	Weekday Afternoon	Weekend Afternoon
Precipitation	0	0.12 inch
Snowfall	0	0
Average temperature	62 $^\circ F$	76 $^\circ F$
AVs in donor's cell	20	91
Average AVs in donor's neighboring cells	40	250
AVs in recipient's cell	30	300
AVs in donor and recipient's cells with vehicle	21	116

Table 1. Two example data points for the predictive model.

GP	1	2	3	4	5
Kernel	DP	DP	Matern	RBF	RBF
Alpha	0.5	0.01	0.3	0.1	0.03

Table 2. GP parameters. Alpha is the dual coefficient of training data points in kernel space. DP means dot product.

All the six models are trained on the same data D_A. We use the mean values of the GPs' predictions and the binary label of the RF classifier, on D_B, as the input to the neural network meta learner. We report the results in Sec. 2.1.3.

2.1.3 Results

We "predict the future with the past". As mentioned at the beginning of Section 2.1, we treat rescues done by volunteers who have done over 10 rescues in the last two months in the last hour of the pick-up window as unclaimed. Thus, we exclude the rescues in the first two months from our prediction task as we do not have the volunteer history for these early entries. As a result, the training data consist of rescues from May 2018 to December 2018 and testing data consist of rescues from January 2019 to May 2019. In addition, since the dataset is imbalanced on the number of claimed and unclaimed rescues, we oversample the unclaimed rescues so that the ratio of claimed and unclaimed rescues is 1:1. The oversampling is applied to only the training dataset for the predictive model.

Table 3 shows the stacking model outperforms all baseline models. Moreover, it yields almost no false positive errors. This is especially important in the food rescue operation, as the cost of not taking actions to a rescue which turns out

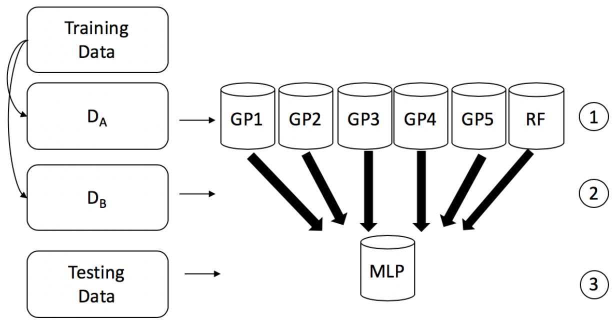


Fig. 3. The stacking model.

Model	Accuracy	Precision	Recall	F1	AUC
GB	0.73	0.86	0.82	0.84	0.51
RF	0.71	0.87	0.78	0.82	0.54
GP	0.56	0.88	0.54	0.67	0.60
SM	0.69	1.00*	0.64	0.78	0.81

Table 3. Performance of selected models, GB: Gradient Boosting Classifier, RF: Random Forest; GP: Gaussian Process; SM: Stacking Model. * We run the experiments for SM for 3 times. The precisions are 1.0, 1.0, 0.9969.

unclaimed due to a false positive is much higher than that of an unnecessary dispatcher intervention due to a false negative.

2.2 Optimizing Intervention and Notification

Our second step is to optimize the intervention and notification scheme (INS) of 412FR, thereby suggesting a guideline for dispatcher intervention and the rules for sending notifications. Our goal is to reduce the frequency that the dispatcher intervenes to "save" a rescue, or the mobile app notifications sent, ideally both.

We formalize the problem by defining an INS as a tuple (x, y, z), with x, y, z described below. When a rescue is posted, the mobile app first sends notifications to the volunteers who are within y miles from the donor. If no volunteer claims the rescue within the first x minutes, the app then sends the notification again to all volunteers who have indicated availability in the corresponding time slot. The dispatcher monitors the rescue after it is posted. If a rescue has not been claimed by z minutes before its pickup deadline, the dispatcher intervenes by directly contacting a group of regular volunteers and asking them if they are willing to claim it. If w_r is the duration from the posting time to the pickup deadline of rescue r, then the dispatcher intervenes $w_r - z$ minutes after the rescue is posted. We assume that upon the dispatcher's intervention, with probability μ the rescue immediately gets claimed, otherwise it has no effect.

412FR has always used a default INS: $\hat{x} = 15$ (minutes), $\hat{y} = 5$ (miles), and $\hat{z} = 60$ (minutes). We look for the optimal INS in a finite set S of candidate

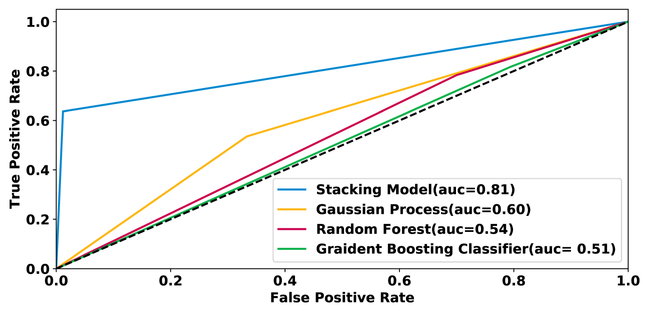


Fig. 4. The ROC curves of the models

Notation	Meaning
x	Second round notification time, default $\hat{x}$
y	First round notification radius, default $\hat{y}$
z	Intervention time from deadline, default $\hat{z}$
μ	Dispatcher intervention success probability
r	r.v.: a rescue, following distribution R.
w_r	Duration from r being posted to deadline
λ	Trade-off of intervention and notification.
$s(\cdot)$	Average number of dispatcher interventions
$v(\cdot)$	Average number of 1st round notifications
$q(\cdot)$	Average number of 2nd round notifications
$p(a, \cdot)$	Proportion of rescues claimed in a minutes
S	Domain of optimization variables (x, y, z)
b_i	Claim rate lower bound

Table 4. Notations for the optimization problem.

INSs which minimizes

$$\lambda \mathbb{E}_{r\sim R}[c_1(x, y, z, r)] + \mathbb{E}_{r\sim R}[c_2(x, y, z, r)] \tag{1}$$

where R is the distribution of rescues, and λ controls the trade-off between two quantities: the expected number of dispatcher interventions and the expected number of notifications sent to volunteers. c_1 is the average number of dispatcher intervention for rescue r, c_2 is the average number of notifications sent for r given the INS. We also want to maintain a high claim rate, i.e., $E_{r\sim R}[c_3(a_i, x, y, z, r)] \geq b_i$ for a given set of a_i, b_i where c_3 is the probability that rescue r is claimed within first a_i minutes.

Without knowing the exact distribution R, we can only estimate these expected values through data. Given a dataset D of rescues under INS (x, y, z), we define $p(a, x, y, z)$ as the proportion of rescues in D that are claimed in a minutes; $s(x, y, z)$ as the proportion of rescues in D that are not claimed by volunteers before the dispatcher intervenes; $v(y)$ as the average number of available volunteers who are within y miles of the donor who receive the first round notifications; $q(x, y, z)$ as the average number of available volunteers who receive the second round notifications. Formally,

$$
\begin{aligned}
p(a, x, y, z) &= \frac{1}{|D|} \sum\nolimits_{r \in D} \mathbb{I}\left(\text{rescue } r \text{ claimed in } a \text{ min}\right), \\
s(x, y, z) &= \frac{1}{|D|} \sum\nolimits_{r \in D} \mathbb{I}\left(r \text{ not claimed in } w_r - z \text{ min}\right), \\
v(y) &= \frac{1}{|D|} \sum_{r \in D} \# \text{ available volunteers within } y \text{ miles of } r \\
q(x, y, z) &= \frac{1}{|D|} \sum_{r \in D} \mathbb{I} \begin{pmatrix} r \quad \text{not} \\ \text{claimed in } x \\ \text{min} \end{pmatrix} \times \begin{matrix} \# \text{ available} \\ \text{volunteers for} \\ r \end{matrix}
\end{aligned}
$$

Assuming data points in D are sampled from R, we have

$$
\begin{aligned}
\mathbb{E}_{r \sim R}[c_1(x, y, z, r)] &\approx s(x, y, z) \\
\mathbb{E}_{r \sim R}[c_2(x, y, z, r)] &\approx v(y) + q(x, y, z) \\
\mathbb{E}_{r \sim R}[c_3(a, x, y, z, r)] &\approx p(a, x, y, z)
\end{aligned}
$$

Our final optimization problem is as follows.

$$
\min_{x,y,z} \quad C(x, y, z) = \lambda s(x, y, z) + v(y) + q(x, y, z) \tag{2}
$$

$$
s.t. \quad p(a_i, x, y, z) \geq b_i, \quad \forall i \in I \tag{3}
$$

$$
(x, y, z) \in S
$$

From the historical data and dispatcher's advice, we could estimate μ, V_y, a_i, b_i, S. However, estimating $s(\cdot), q(\cdot)$, $p(\cdot)$ poses significant difficulty. We need to estimate the counterfactual claim time (CCT) for all INSs $(x, y, z) \neq (\hat{x}, \hat{y}, \hat{z})$.

2.2.1 Counterfactual Claim Time (CCT) Estimation

Given a rescue happened under the default INS $(\hat{x}, \hat{y}, \hat{z})$, we estimate its CCT under some other INS (x, y, z). We make the following assumptions.

- No matter when a volunteer receives the notification, upon receiving it they take the same amount of time to respond, and the effect of human intervention is independent of the app notification.
- The intervention outcome is not affected by the INS.
- Given a list of regular volunteers (provided by dispatchers or derived from data), if a rescue is recorded in the historical data as claimed by a regular volunteer after the dispatcher intervention time, i.e., $w - \hat{z}$ minutes after the rescue is posted, we give the credit to dispatcher intervention. If a rescue was claimed after the dispatcher intervention time by anyone else, we assume that the dispatcher's intervention have failed.

Dispatcher intervention	Notification radius	Distance	Counterfactual claim time
$a < w_r - \hat{z}$	$y \leq \hat{y}$	$d \leq y$	$m_z(a)$
		$d \in (y, \hat{y}]$	$m_z(a + x)$
		$d > \hat{y}$	$m_z(a + x - \hat{x})$
	$y > \hat{y}$	$d \leq y$	$m_z(a)$
		$d \in (\hat{y}, y]$	$m_z(a - \hat{x})$
		$d > y$	$m_z(a + x - \hat{x})$
$a \geq w_r - \hat{z}$, by regular volunteer	any	any	z
$a \geq w_r - \hat{z}$, by other volunteer	$y \leq \hat{y}$	$d \leq y$	a
		$d \in (y, \hat{y}]$	$a + x$
		$d > \hat{y}$	$a + x - \hat{x}$
	$y > \hat{y}$	$d \leq y$	a
		$d \in (\hat{y}, y]$	$a - \hat{x}$
		$d > y$	$a + x - \hat{x}$

Fig. 5. Construction of the CCT for INS (x, y, z) based on default INS $(\hat{x}, \hat{y}, \hat{z})$. a is the rescue's actual claim time. d is the distance from the rescue's volunteer to the donor.

Suppose the rescue was claimed by volunteer i located d miles from the donor in the historical data. At a high level, in most cases we compute the claim time of volunteer i in the new INS (x, y, z) and take that as our CCT estimate. For example, suppose i is within the first round notification radius, i.e. $d \leq \hat{y}$ and claims the rescue in 7 minutes under $(\hat{x}, \hat{y}, \hat{z})$. This rescue would have a CCT of 12 minutes when $x = 5, z = \hat{z} = 60$ and $y < d \leq \hat{y}$, i.e., i is now outside the first round notification radius. This is because the volunteer i needs 7 minutes to respond after getting notification, but now they only receive the notification 5 minutes after the rescue is available. We also factor in the effect of dispatcher intervention when the intervention happens before the CCT k, i.e. $w_r - z < k$. For rescue r, we report the expected claim time $m_z(k) = \mu \min\{w_r - z, k\} + (1 - \mu)k$. In another scenario, if in the historical data, volunteer i who is not in the first round notification radius claims the rescue before the second round notification, we assume the volunteer's action is due to actively checking the available rescues and is not affected by the notification. Thus, the CCT remains the same for all INS. The complete computation is shown in Fig. 5.

Our estimation is conservative, i.e., we will never underestimate the claim time. This is important in practice, because overestimation may merely lead to unnecessary resource spent but underestimation may cause a rescue to fail. Our estimation is accurate when i is within the first round notification radius in the counterfactual INS but not in the default INS and intervention happens after the claim time, as i would still be the first volunteer to claim the rescue under the counterfactual INS. In some other cases, there exists the unobservable possibility that some other volunteer might claim the rescue before i in the counterfactual INS, and hence we might overestimate the claim time.

$s(x,y,z)$	x_{min}	y_{max}	z_{min}
$v(y)$		y_{min}	
$q(x,y,z)$	x_{max}	y_{max}	z_{max}

Table 5. Replace (unspecified) variables in each term with the extreme values to get a lower bound.

2.2.2 Solving the Optimization Problem

Given the CCT estimate for each rescue, we can estimate the functions $s(\cdot), q(\cdot), p(\cdot)$ using the counterfactual dataset. However, there is no closed-form expression for them. Computing their values at every point in a brute force way is obviously inefficient. We propose a branch-and-bound algorithm and a feasibility check to find optimal INS more efficiently.

First, we note that the CCT, as detailed in Fig. 5, is increasing in x and decreasing in y and z. Since $p(\cdot)$ is the empirical estimate based on the claim time, if some infeasible INS (x, y, z) does not satisfy claim rate constraint (3), any INS $(\tilde{x}, \tilde{y}, \tilde{z})$ with $\tilde{x} \geq x, \tilde{y} \leq y, \tilde{z} \leq z$ is also infeasible. Thus, we need not generate CCT for $(\tilde{x}, \tilde{y}, \tilde{z})$.

Using a similar observation, we devise our main algorithm, Alg. 2. Note that $s(x, y, z)$ decreases as x, z decreases and y increases, $v(y)$ decreases as y decreases, $q(x, y, z)$ decreases as x, y, z increases. Therefore, if we replace all the variables in all terms with the extreme values in domain S that can minimize $C(x, y, z)$ (as shown in Table 5), we get a lower bound of $C(x, y, z)$. We define a subproblem as the original optimization problem with k of the variables in the INS specified and the remaining ones unspecified for $k = 0, 1, 2, 3$. To compute a lower bound for each subproblem, we replace the unspecified variables in each term with the extreme values according to Table 5. For example, if z is specified, and x, y are unspecified, we get a lower bound

$$\bar{C} = \lambda s(x_{min}, y_{max}, z) + v(y_{min}) + q(x_{max}, y_{max}, z)$$

In Alg. 2, we start with the original problem where none of the variables are specified ($k = 0$). We branch to lower level subproblems in the order of $z \rightarrow y \rightarrow x$, as this order tends to prune the fastest. For each subproblem, we either compute a lower bound, or when all variables are specified, compute the exact cost. We generate one counterfactual dataset for computing the exact cost (Line 3, Alg. 1), and at most two datasets when computing the lower bound (Line 8, Alg. 1), since $s(\cdot)$ and $z(\cdot)$ are minimized at two different INSs and $v(\cdot)$ does not depend on the CCT. The implicit pruning on Line 3 guarantees Alg. 2 finds the optimal solution.

2.2.3 Results

After consulting the dispatchers, we take $\mu = 0.4$ as the probability that dispatcher intervention is effective. We require that the optimal INS's claim rate

Algorithm 1: Solve-Relaxation

```
Optional input arguments: $x, y, z$
if all of $x, y, z$ specified then
    Generate CCTs with $(x, y, z)$.
    if feasible then
        Compute cost $\bar{C} = C(x, y, z)$
        return subproblem $(\bar{C}, (x, y, z))$
else
    Generate CCTs with unspecified parameter replaced by extreme values in
      Table 5.
    Compute lower bound $\bar{C}$
    return subproblem $(\bar{C}, (x, y, z))$
```

Algorithm 2: Branch-and-Bound for INS

```
Push Solve-Relaxation({}) to Frontier.
while Frontier set is not empty do
    Get subproblem with lowest $\bar{C}$ from Frontier.
    if subproblem has all parameters specified then
        return $(\bar{C}, (x, y, z))$   // (optimal solution)
    else
        Follow the order $z \to y \to x$ to expand the node, i.e., if the first $k$
          variables are already specified, create a subproblem for each possible
          value of the $(k+1)^{th}$ variable in $S$.
        Add all subproblems Solve-Relaxation($x, y, z$) to Frontier.
```

be no worse than default INS. That is, we use $a_i = 1, 2, \ldots 120$ and b_i being the empirical claim rate at the a_i-th minute under the default INS.

First, we demonstrate the effectiveness of the branch and bound algorithm (Alg. 2). We set the domain S as $x, y \in \{2, 4, 6, 8\}$ and $z \in \{30, 40, 50, 60\}$. As shown in Table 6, branch and bound needs to generate CCTs on much less INSs than the brute force approach, although advantage is less significant for smaller λ. In the sequel, we use Alg. 2 and set the domain S as $x \in \{1, 1.5, 2, \ldots, 25\}, y \in \{1, 1.5, 2, \ldots, 10\}, z \in \{30, 32.5, 35, \ldots, 90\}$.

Similar as above, we use the earlier data D_{past} to predict the more recent data D_{future}. First, we focus on computing the optimal INS on D_{past} in Fig. 6a. Both the 2nd round notification time and the dispatcher intervention time decrease as λ grows, i.e. the dispatcher's intervention matters more in the dispatching cost. This is aligned with the results in Table 5. When the app notification is the primary concern, the default INS is almost desirable, yet if we would like to minimize the interventions, the 2nd round notification needs to go out sooner. Fig. 6b, we show the Pareto frontier (in red) of optimizing on D_{past}. The optimal INSs in Fig. 6a are now shown in blue. The default INS lies within

	Brute force search		Branch and bound	
λ	INSs	Time (s)	INSs	Time (s)
10^7	64	192.6	18	65.5
10^6	64	183.7	18	64.9
10^5	64	185.1	16	56.4
10^4	64	190.9	34	125.0
10^3	64	187.1	35	129.3

Table 6. Running time and the number of INSs for which the CCTs are generated.

INS	Interventions	Notifications
A: $(16.5, 5.5, 45)$	$-13\%\,(-0.06)$	$0\%\,(-1)$
B: $(15.5, 5.5, 32.5)$	$-24\%\,(-0.10)$	$+2\%\,(+46)$

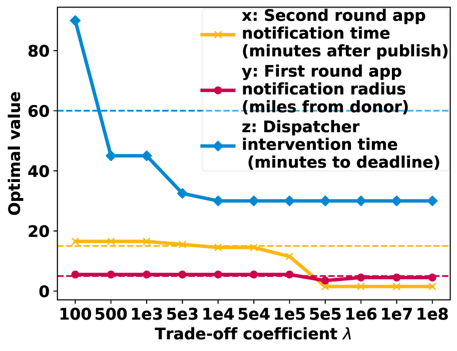
Table 7. The projected change in the probability of interventions and number of notifications of the proposed INSs. The numbers in parentheses are the absolute change.

the frontier, suggesting that the numbers of both interventions and notifications can be improved. The orange rectangle indicates the INS region that is strictly superior to the default INS.

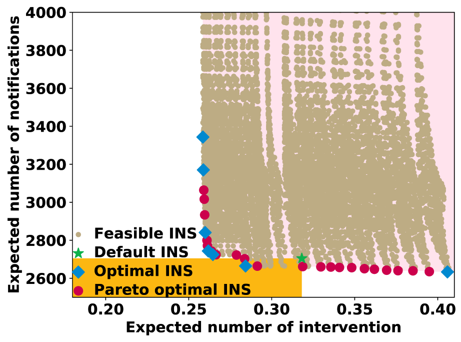
Of course, we would like to examine the quality of the optimization solutions on unseen data. Thus, in Fig. 6c, we show the projected number of interventions and notifications on D_{future} of the optimal INSs on D_{past}. For the same INS, the performance is different between Fig. 6b and Fig. 6c because the claim probabilities are estimated using the two datasets separately. Despite this difference, some optimal INSs on D_{past} still outperforms the current practice. We therefore suggest two INSs (see Fig. 6c) to 412 Food Rescue, as shown in Table 7. INS A is a strict improvement over the current practice, reducing the number of both intervention and notification. Our second solution, INS B, drastically reduce the labor of dispatcher by 24% at the expense of a mere 2% increase of notifications sent. Since 412FR handles 4574 rescues in a 430-day period, INS B can save the dispatcher over 390 times of intervention a year in expectation. An intervention takes the dispatcher at least the same amount of time as matching a new food rescue, and often more. Thus, the dispatcher could handle at least 390 extra rescues a year, which is over 7500 pounds of food by the average donation in our dataset. We choose INS B over the rightmost INS on Fig. 6c because 412FR has relatively more shortage of dispatcher than volunteers. Finally, Fig. 6d shows our two INSs have competitive claim rates on the unseen data. This suggests the promise of deploying our method in the future.

2.3 Rescue-Specific Push Notifications

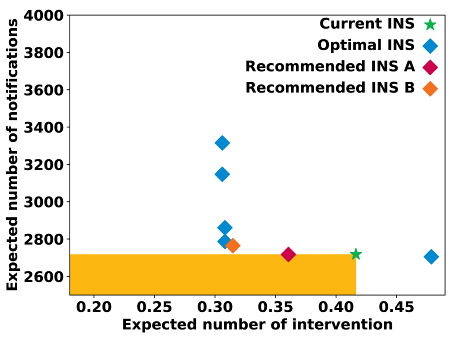
In Section 2.2, we introduced a generic distance-based push notification scheme that would apply to all rescues. Although the distance between the volunteer and the pick-up location would clearly matter, it is reasonable to consider if other factors would also affect the chance that a volunteer would claim a rescue.



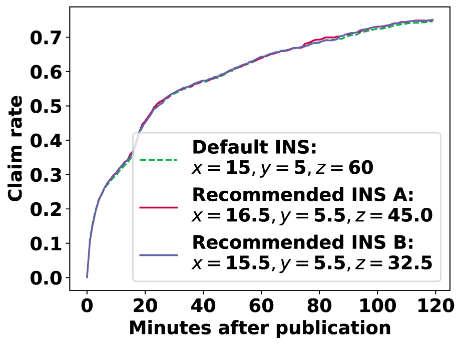
(a) Optimal INSs on D_{past}



(b) Pareto frontier on D_{past}



(c) Performance of the optimal INSs for D_{past} on D_{future}



(d) Projected claim rate of two recommended INSs on D_{future}

Fig. 6. Experiment results of the data-driven optimization

This leads to tailoring push notifications to each rescue. That is, for each rescue, we select a subset of volunteers to send push notifications to. In this section, we study this as a recommender system problem.

2.3.1 Data

To develop a recommender system, we need both positive and negative labeled examples. A positive example means that a particular volunteer (item) claims a particular rescue (user); a negative example means otherwise. In this section, we detail our data acquisition, labeling, and feature engineering process.

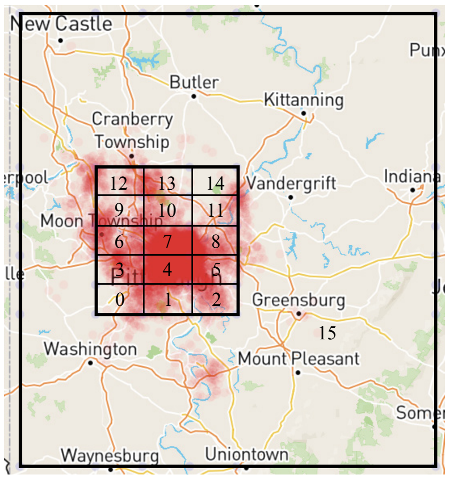
Positive Labels We use the rescue database from 412FR, covering the period from March 2018 to March 2020. The database keeps the log of each rescue. For most rescues that have been claimed and completed by a volunteer, we simply take the rescue plus the volunteer who claimed it as a positive data point. However, the food rescue operation is not always so neat. Occasionally, the dispatcher knows ahead of time that some volunteer would do the job, so they directly assign the volunteer for a particular rescue and bypass the app notification stage. In this case, we take this direct assignment as a positive example as well. Sometimes a volunteer might claim a rescue and then drop it, causing some rescue to have multiple volunteers in the log. In this case, we create our labels based on the last volunteer.

Negative Labels A negative example means that a particular volunteer did not claim a particular rescue. Since almost all rescues have only one volunteer who claimed the rescue, obviously most of our data points will have negative labels. However, not all of these negative data points are necessarily true, because perhaps a volunteer would have claimed some rescue if someone else had not claimed it 10 minutes in advance. Thus, we use the following ways to construct a selected negative dataset. First, recall that 412 Food Rescue used a mobile app push notification scheme which notifies volunteers within 5 miles when the rescue is first available and then notifies all volunteers 15 minutes later if the rescue has not been claimed. Thus, if a rescue is claimed within 15 minutes, we only treat the volunteers who were within 5 miles and did not opt out of push notifications as negative examples.

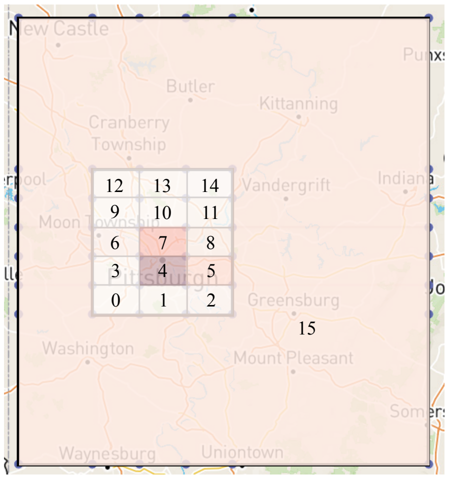
We also incorporate another data source to strengthen our negative sampling. In addition to mobile app notifications, recall that the dispatcher at 412FR also manually call some regular volunteers to ask for help with a specific rescue. We obtained the call history from 412FR, from which we identify the volunteers they reached out to within the time frame of each rescue. If these volunteers did not claim the rescues in the end, we treat them as negative examples. Compared to the negative examples derived from push notifications, we have more confidence in this set of negative examples, since declining on a phone call is a stronger indicator than ignoring a push notification.

Feature Engineering We carefully identify a selected set of useful features that are relevant in the food rescue operation.

First, the experience of food rescue dispatcher indicates that if a volunteer has completed a rescue at or near a donor or recipient, they are more likely to do a rescue trip again in the neighborhood. As shown in Figure 7, we divide the Greater Pittsburgh Region into 16 cells. We evenly divide a central rectangular region into a 3×5 grid, and label them grid cells 0 through 14. Then, we label the entire map outside the rectangular region cell 15. The rationale is that in the outer suburbs there are fewer donors, recipients, and volunteers, and furthermore volunteers who in suburbs are more willing to do long-distance, i.e. inter-cell, rescue than volunteers in downtown. For each rescue trip and each volunteer, we calculate the number of rescues the volunteer has done in the rescue donor's cell, in the rescue recipient's cell, and across all cells. These counts are only up to the date of the given rescue, so that we could prevent data leakage. We also tried to include as features the volunteer's historical rescues in each cell, not just the donor's and recipient's cell. However, they did not contribute any predictive power and thus we leave them out of the final model.



(a) Distribution of donor organizations. Darker colors mean more frequent donations. We plot the donor locations with random perturbations.



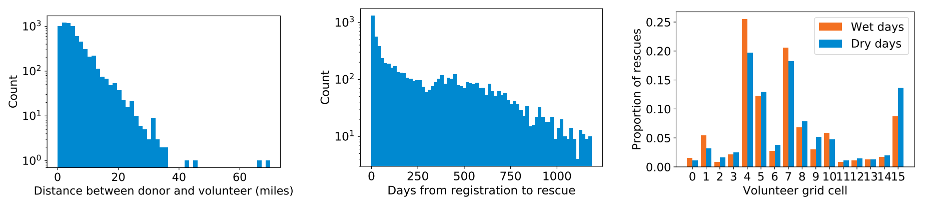
(b) Density of recipient organizations. Darker colors mean more recipient organizations in the grid.

Fig. 7. We divide the Pittsburgh area into 16 grid cells, with cells 0–14 covering downtown Pittsburgh and its neighborhoods, and cell 15 containing the rest of the region.

Closely related to this is the distance between the volunteer and the donor. It is unlikely that a volunteer would drive 30 miles to pick up a donation, as we show in Figure 8a. We measure the distance using the straight line distance based on geographic coordinates. Although the actual traveling distance might be a better indicator, we observe that the straight line distance already serves our purpose.

Aside from the geographical information, the length of time between volunteer's registration on the platform and the rescue is also an important factor, as suggested by our collaborators at 412FR. We plot the histogram of this variable in Figure 8b. Immediately after registration, the volunteer is eager to claim a rescue to get a feel of the food rescue experience. If a volunteer has stuck with the program for an extended period and remains active, it is likely that they are a regular and dependable one as well, which is substantiated with the upward trend and plateau in Figure 8b around days 300–600. Thus, we include this feature in our prediction model.

Weather information is also an important factor in the prediction. Presumably rainy and snowy days would see a lower volunteer activity in general. However, the impact of inclement weather would fall disproportionately on volunteers who do not have a car or live in suburban areas. Same as in Section 2.1, we use the Climate Data Online (CDO) service provided by the National Oceanic and Atmospheric Administration to access the weather information. The CDO dataset



(a) Histogram of rescues, based on the distance between the donor and the volunteer who claimed the rescue.

(b) Histogram of rescues, based on the length of time between the rescue and the registration of the volunteer who claimed the rescue.

(c) Histograms of rescues under wet and dry weather, based on the location of the volunteer who claimed the rescue.

Fig. 8. Data analysis results.

Layer	Operation	Hidden Units
1	Dense (ReLU)	384
2	Dense (ReLU)	2048
3	Dense (ReLU)	512
4	Dense (Logistic)	16

Table 8. Neural network architecture

contains weather information at the discretization level of days and weather station. There are multiple weather stations in the Pittsburgh area and for each rescue we select the data for the date of rescue and the station that is closest to the donor organization. As shown in Figure 8c, on wet days, relatively more volunteers who claim the rescue reside in downtown Pittsburgh (cell 4 and 7). Whereas on dry days, a lot more volunteers who live in the outer suburbs of Pittsburgh (cell 15) are active. In fact, we also saw a significant difference in the average distance between volunteer and donor for dry days (5.94 miles) and rainy days (5.22 miles), with a t-test p-value 3×10^{-8}.

We also explored a number of other features but did not incorporate them into our final model. These features include the rescue's time of day and day of week, the volunteer's availability, whether the volunteer uploaded an avatar to their profile or not, whether the volunteer is located in the same grid as the donor or recipient, and so on. Although these are intuitive factors, we did not find them improve the predictive power of our model and hence left them out.

2.3.2 Recommender System

We build a neural network-based recommender system. We first detail our network architecture, and then discuss our approaches to address the unique challenges in the food rescue domain.

We show the neural network architecture in Table 8. The input to the neural network is the feature vector of a rescue-volunteer pair. The feature vector passes through four dense layers. Each layer is followed by a ReLU activation function, except for the last layer where we output a single number which is then converted to a number between 0 and 1 by the logistic function. This output represents the likelihood that this volunteer will claim this rescue trip. We use the cross entropy loss to train the neural network. To output a list of k volunteers to whom we send push notifications for a particular rescue at prediction time, we pass the feature vectors of the rescue-volunteer pairs for all volunteers on a fixed rescue through the network and rank the output to take the top k of them.

Negative Sampling As mentioned earlier, there is an extremely high label imbalance in our dataset. From March 2018 to March 2020, there are 6757 rescues available for the training. Each rescue typically has only one volunteer who claimed it, and there are 9212 registered active volunteers in the Pittsburgh area. This means, theoretically, the ratio between negative and positive examples is over $9000 : 1$. Using the method introduced in Section 2.3.1, we can obtain a selected set of negative examples D_n derived from push notifications and another set of negative examples D_c derived from dispatcher calls. The set D_c is slightly smaller than the positive examples D_p, while $|D_n| : |D_p| \approx 700 : 1$. When training the neural network, we always use all the examples from D_p and D_c. However, we randomly sample a subset of examples from D_n at each episode of the training. By doing this, we ensure that the negative examples from D_n do not dominate the training set, and at the same time the "more certain" negative examples from D_c gets emphasized more than D_n. This whole procedure leads to an overall ratio between negative and positive samples around $3 : 1$ in each single batch.

Diversity and Online Planning Recommender systems in general suffer from the diversity issue, where "hot" items get recommended to all the users. In commercial applications, this might lead to the "rich gets richer" phenomenon on superstar items and the missed revenue opportunity on the less popular items. All these are valid. However, as we have emphasized several times in this paper, the "items" on the other side of our recommender system are humans. The aforementioned consequences of the lack of diversity is only going to be more problematic in our case. If a popular volunteer received push notifications for every single rescue throughout the day, they would possibly get annoyed and mute the notifications. On the other hand, for volunteers who are already not very active, if our system never sent them push notifications, they would probably just forget about the platform and would be unlikely to return. Therefore, it is crucial that we properly handle the diversity issue.

We distinguish between two notions of diversity: individual diversity and aggregate diversity. The former means that each user (rescue) gets recommended a diverse set of items (volunteers). The latter means that the recommended items (volunteers) across different users (rescues) combined cover a large portion of the item space. Our human-centric approach determines that we focus on aggregate diversity here. In fact, we focus on a slightly different metric: how many times each volunteer gets recommended for a rescue every day. We wish to put a cap on this metric, which is directly linked to the user experience of each volunteer.

To this end, we can formulate the following mathematical program for a given day of food rescue operation.

$$
\begin{aligned}
(\mathit{\Pi}) \quad \max_{x} \quad & \sum_{i\in R}\sum_{j\in V} p_{ij}x_{ij} \\
s.t. \quad & \sum_{j\in V} x_{ij} \le k, \quad \forall i \in R \\
& \sum_{i\in R} x_{ij} \le b, \quad \forall j \in V \\
& x_{ij} \in \{0,1\}, \quad \forall i \in R, \forall j \in V
\end{aligned}
$$

Let V denote the set of volunteers. On a particular day, we have a set of rescues R. The binary decision variable x_{ij} is equal to one if we decide to send push notification to volunteer j about rescue i. The first constraint indicates that for each rescue we will notify the top k volunteers, as introduced at the beginning of Section 2.3.2. The second constraint is our diversity constraint, which makes sure that each volunteer receives at most b push notifications a day. The p_{ij} in the objective is the output from our trained neural network, representing the predicted likelihood that volunteer j is going to claim rescue i.

While this optimization problem $\mathit{\Pi}$ is a valid method to improve diversity in generic recommender systems, it does not solve the problem in our setting. The reason is that donations, and hence food rescue trips, arrive in our system sequentially throughout the day, and the dispatcher must also act in real-time. It is unacceptable to wait till the end of the day, run the optimization problem above, and then send the push notifications. Therefore, we need an online algorithm.

An intuitive approach is to resort to the literature on online linear programming [1]. Indeed, we could imagine solving π where at each time step, a new rescue is revealed with a new column in the x matrix and p matrix. However, we do not know how many rescues there will be at the beginning of the day. This is a major obstacle in applying the established algorithms with theoretical guarantees. Instead, the daily rescue pattern is hardly adversarial in nature and thus we propose a simple heuristic, as shown in Algorithm 3.

In Algorithm 3, when a food rescue arrives in the system, we sample the historical rescue data for trajectories. Typically, we would sample the rescues on the same weekday a week ago, two weeks ago, and so on. The underlying idea is that the same weekdays might have similar rescue patterns. This is because most donations that come to 412FR come from grocery stores or large companies/universities. Grocery stores often perform inventory counts on a weekly basis. Companies and universities often hold weekly events, with catered food. For each sampled day, we only take the trajectory from the time of the current rescue to the end of the day. Then, for each trajectory along with the current

Algorithm 3: Online Planning for Optimizing Push Notifications

```
1 A trained neural network predictor while $a$ $new$ $rescue$ $i$ $arrives$ do
2     Flush $X_i$
3     for $dayToSample = 1, 2, \ldots H$ do
4         Sample the set of rescues $R$ on the dayToSample that occured from
            the time of the current rescue $i$ till the end of the day.
5         Compute predicted claim probabilities $p_{ij}$ and $p_{i'j}$ for all $i \in R$, for all
            $j \in V$.
6         Solve the following optimization problem:
              $$\begin{aligned} (\Pi_i) \quad \max_x \quad & \sum_{j \in V} \left( p_{ij} x_{ij} + \sum_{i' \in R} p_{i'j} x_{i'j} \right) \\ s.t. \quad & \sum_{j \in V} x_{i'j} \leq k, \quad \forall i' \in R \\ & \sum_{j \in V} x_{ij} \leq k \\ & x_{ij} + \sum_{i' \in R} x_{i'j} \leq b_j, \quad \forall j \in V \\ & x_{ij} \in \{0, 1\}, \quad \forall i \in R, \forall j \in V \end{aligned}$$
7         Keep in $X_i$ the optimal solution $x^*_{ij}$ for the current rescue only.
8     Sum $X_i$ over the sampled histories, find the top $k$ volunteers.
9     Send push notifications to them. Update the remaining budget $b_j$ for each
        volunteer $j$.
```

rescue, we obtain the neural network's predicted claim probabilities and solve the optimization problem Π_i. Π_i is similar to Π except that each volunteer now has their own remaining budget of push notification. Note that now everything in Π_i is observed and known, whereas in Π the future rescues are unknown at the decision-making time. We keep only the part of the optimal solution that concerns the current rescue and discard the rest. Later, on Line 8 in Algorithm 3, for each volunteer, we sum over its value in the optimal solution across all the sampled trajectories. We take the top k volunteers as voted by these solutions, who become the ones we will send push notifications to for this current rescue.

We note that the optimization problem Π and Algorithm 3 are extremely flexible to account for many additional considerations. For example, we could use personal budget b_j in Π and add additional constraints to represent the volunteer's push notification preferences. We could also add weights to the objective function to emphasize the importance of a particular rescue.

2.3.3 Results

Recommender System We use a training set containing rescues from March 2018 to October 2019, which is 80% of the entire dataset. We use the remaining 1373 rescues from November 2019 to March 2020 as the test set.

First, we only consider the prediction part of our algorithm. We compare our neural network recommender system with several competitive baselines that are commonly used, including random forest (RF), gradient boosted decision trees (GBDT), and stacking model (SM). To determine the hyper-parameters of the baseline models and the neural network model, we separate a validation set which consists of the last 1/8 of our training set and then run a grid search according to the performance on the validation set. For experiments on the baselines, we use the same negative sampling method on D_c and D_p as described in Section 2.3.2. As for negative examples from the app notifications D_n, since the baselines are not gradient descent-based methods, we sample them in two schemes such that the ratio between the positive and negative examples is roughly 1 : 1 and 1 : 20, respectively. We consider the latter because that is roughly the number of negative examples that the neural network approach has seen throughout the training, in order to ensure a fair comparison.

We show the results of all these algorithms on the test set, averaged over 5 runs, in Table 9. We consider the hit ratio at k (HR@k) and the normalized discounted cumulative gain at k (NDCG@k) in Table 9. However, we note that our main metric of interest is the hit ratio, because when sending push notifications, we do not care about the particular order in which each volunteer ranks on the list. Also because of this, HR@k is our primary metric during the grid search on hyper-parameters for all the predictive models. We choose the value of k to be 964, since this is the average number of push notifications sent per rescue under the current notification scheme. The current distance-based notification scheme has a hit ratio of 0.4392. All baselines show better performance than the current method, with random forest and GBDT being better than the stacking model. However, the neural network based prediction model outperforms all the baselines.

The hit ratio of the neural network model is a 66% improvement over that of the current distance-based method. This means that we would be able to reach the would-be volunteer in approximately 900 more rescues every year. Each of these rescues has a donor and a recipient organization that serves tens or hundreds of people behind it. A smooth food rescue experience would not only provide basic food necessities to these people, but also encourage these organizations to keep up the engagement in a sustainable way.

Diversity and Online Planning As mentioned in Section 2.3.2, recommender systems, in general, suffer from the diversity issue. This problem also exists in our model. In Figure 9, we plot the histogram of the number of push notifications received by each volunteer for the test set rescues. The neural network based recommender system, shown in yellow in Figure 9, exhibits an alarming bimodal distribution: most volunteers either receive almost no push notifications, or re-

Model	HR@k (SD)	NDCG@k (SD)
NN	**0.7269 (0.0310)**	**0.1898 (0.0147)**
RF(1:1)	0.5989 (0.0395)	0.1319 (0.0303)
RF(1:20)	0.6035 (0.0511)	0.127 (0.0053)
GBDT(1:1)	0.6235 (0.0549)	0.1613 (0.0098)
GBDT(1:20)	0.5394 (0.0152)	0.1023 (0.0086)
SM(1:1)	0.4996 (0.0005)	0.1332 (0.0002)
SM(1:20)	0.5219 (0.0125)	0.0948 (0.0030)
Default	0.4392 (N/A)	N/A (N/A)

Table 9. The performance of the neural network based recommender system and several baselines. All experiments are repeated five times with the mean and standard deviation shown in the table.

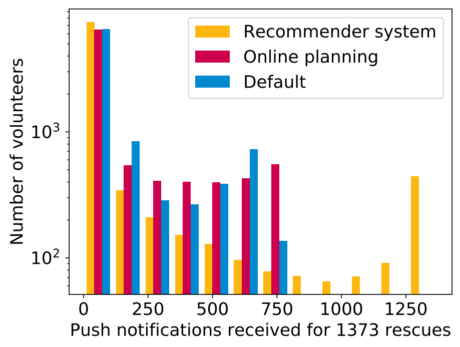


Fig. 9. Histograms of the number of push notifications received by each volunteer over all the 1373 rescues in the test set. The online planning algorithm has a budget of 6 notifications per day.

ceive push notifications for almost every single rescue. We remark that although the number of volunteers in the rightmost bin (446 out of 9312) is much smaller than that in the leftmost bin (7458 out of 9312), the former is much more concerning. This is because they are typically the most "active" volunteers who have contributed the most to the food rescue program. In fact, these 446 volunteers contain 39 of the top 50 most frequent volunteers, and 51 of the top 100. If they left the platform due to too many notifications, which is likely to happen should the proposed recommender system get deployed, the loss to 412FR would be disproportionately high. On the other hand, the default distance-based notification scheme does not suffer from this issue, as shown in red in Figure 9. Although the majority of the volunteers still receive few push notifications, the notification frequency for each volunteer is capped at roughly once every two rescues.

Figure 9 serves as a stern warning against the premature deployment of machine learning algorithms in the real world. That a certain model outperforms the current practice by 66% in some important metric (here, the hit ratio) does not mean it would not cause other problems.

We use our Algorithm 3 to improve the diversity of volunteer recommendations. As a preliminary and straightforward comparison, we ran our online planning Algorithm 3 with budget $b = 6$ push notifications per day using the rescues seven days ago as the sampled history. We plot its notification histogram in yellow in Figure 9. It is easy to see that the online planning algorithm achieves a push notification distribution much more similar to the default scheme, than the recommender system alone. It completely avoids sending push notifications about every single rescue to any particular volunteer.

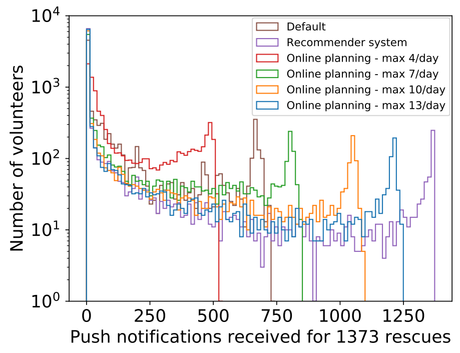


Fig. 10. Histograms of the number of push notifications received by each volunteer over all the 1373 rescues in the test set, compared across different budget values.

Indeed, the effect of Algorithm 3 on recommendation diversity depends on the budget parameter b. In Figure 10, we plot the notification distributions for different choices of the budget value, and compare them against those of the recommender system and the default notification scheme. As the budget increases, the distribution of push notifications from Algorithm 3 approaches that of the recommender system. We note that the position of the rightmost peak of each histogram should not be interpreted as an indicator of the total number of push notifications sent. In all of these experiments, we limit the number of notifications for each rescue at $k = 964$. Except for when the budget is extremely small, the algorithm always notify exactly 964 volunteers for each rescue. The diversity goal here is to make the histogram occupy as little space as possible on the right side of the figure.

Much as we demonstrate the improvement of recommendation diversity, we would also like to ensure that the recommendation accuracy of our algorithm does not drop too much. The budget parameter b captures the inherent tradeoff between diversity and accuracy. As we show in Figure 11, the yellow curve represents the hit ratio of Algorithm 3. Algorithm 3 outperforms the existing notification scheme when the budget is more than four notifications per day, which is a relatively trivial amount. When the budget rises to 10 notifications per day or more, the hit ratio is very close to the bare bone recommender system.

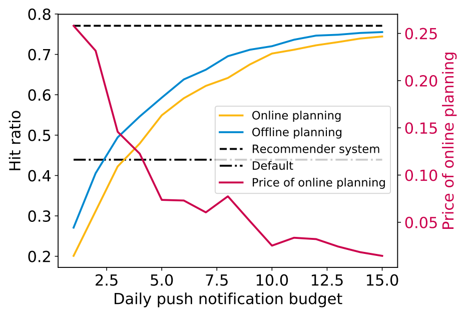


Fig. 11. Hit ratio of the online planing algorithm. Price of online planning, computed as $1 - \frac{HR_{\text{online}}}{HR_{\text{offline}}}$, is shown on the right axis.

In order to further evaluate the quality of *online* planning in Algorithm 3, we also solve an offline version of the problem, where we solve the mathematical program Π separately for each day, assuming full information about the rescues on that day. We show the hit ratio of the recommendation decision from this offline version in blue in Figure 11. Since having full information is always better, the blue curve always lies above the yellow curve representing the online planning. However, the difference is not big. We term the difference as the "price of online planning", which is computed as $1 - \frac{HR_{\text{online}}}{HR_{\text{offline}}}$. In fact, Figure 11 shows that the price of online planning is decreasing as the budget grows, and is consistently smaller than 0.1 when the algorithm is of potential deployment interest (performing better than the current practice). This validates our earlier claim in Section 2.3.2 that the rescues on the same weekday of the previous week are a reasonably good indicator of the rescues on the present day.

3 Resource Requirements

Dataset The food rescue dataset is provided by 412FR under a data use agreement. The weather data are readily available at Climate Data Online.

Computing Resources We conducted all of our experiments on an Intel i7-7700K 4.20GHz CPU with 64GB RAM.

Software Tools We implemented the machine learning components of the code with PyTorch and Scikit-learn. The online planning algorithm requires the integer programming solver Gurobi, though open source substitutes exist as well.

Deployment and Maintenance The recommended INS and the recommender system push notification system are integrated into 412FR's proprietary food rescue management system FoodRescueX.

Condition	Claim Rate	Average time from publish to claim (min)	Average # push notifications sent
Before 2/10/2020 (Previous scheme)	0.84	78.43	11499.45
2/10/2020 - 3/1/2020 (New scheme)	0.88	43.05	9167.52
After 3/1/2020 (After COVID)	0.92	39.73	9735.54

Table 10. Food rescue metrics before and after the adoption of the recommended INS.

4 Field Evaluation

The intervention and notification scheme in Section 2.2.3 has been adopted by 412FR since February 2020. Table 10 shows the key food rescue metrics before and after the adoption. Because COVID-19 started to impact the life in Pittsburgh in March 2020, we split the period after adoption into before COVID and after COVID in order to provide a more objective picture of the results. As shown in Table 10, after the adoption of our recommended INS, the rescue claim rate went up, the rescues got claimed faster, and 412FR had to send fewer push notifications. This indicates that all these three key metrics improved in the desirable direction. After COVID hit the area, the improvement stuck around, and got even better in both the claim rate and the claim speed. Nevertheless, we need to emphasize that this is not a controlled experiment. There could be lots of confounding factors.

We are working with 412FR to deploy the recommender system push notification scheme introduced in Section 2.3. We are launching a randomized control trial to rigorously evaluate the effect of the machine learning-based push notification system. We have integrated the ML model into 412FR's food rescue management platform "FoodRescueX". If the pilot study shows promising results, the ML model could be readily used and maintained in the long term.

5 Lessons Learned

Collaboration between nonprofit organizations and academic researchers is challenging. In what follows, we attempt to gather some lessons we learned in this collaboration that are actionable for academic researchers.

Academic researchers come into these collaborations with a certain toolkit. They have every incentive to drive the collaboration towards a direction that would best fit their research agenda. However, this risks them ending up not working on something the nonprofit organization wants the most. This is disrespectful to their nonprofit collaborators. And they would pay the price eventually – because the nonprofit does not need it, they probably would not want to deploy it. The simple and obvious way out, we believe, is real listening. At the beginning of our collaboration with 412FR, we had a long list of research questions that would leverage our technical expertise. However, the collaboration only took off when we really listened to their needs and threw our own research agenda off the table, even if that meant we had to explore a new research area.

After throwing our research agenda out of the room, we had to throw ourselves into the problem setting. The several meetings with our food rescue partners gave us an initial understanding of the problem setting, but to really understand the problem, we had to be part of that problem as well. We completed food rescue trips ourselves as volunteers, sat with the food rescue dispatchers in the office to observe how they work, and went on rides with the food rescue truck drivers to meet the communities they serve. Such personal experience helped a lot in our later problem formulation and research.

On problem formulation, it is nothing new, but probably worth reiterating, that technology amplifies existing initiatives rather than create new ones [6]. Push notifications and dispatcher interventions are something that FRs had already been doing prior to our work. Thus, our work simply suggests new parameters that FRs may reference in their standard procedure. This made our work more easily accepted and deployed.

References

1. Adomavicius, G., Kwon, Y.: Optimization-based approaches for maximizing aggregate recommendation diversity. INFORMS Journal on Computing **26**(2), 351–369 (2014)
2. Coleman-Jensen, A., Rabbitt, M.P., Gregory, C.A., Singh, A.: Household food security in the united states in 2017. USDA-ERS Economic Research Report (2018)
3. Conrad, Z., Niles, M.T., Neher, D.A., Roy, E.D., Tichenor, N.E., Jahns, L.: Relationship between food waste, diet quality, and environmental sustainability. PloS one **13**(4), e0195405 (2018)
4. Felt, A.P., Egelman, S., Wagner, D.: I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices. pp. 33–44. ACM (2012)
5. Laborde, D., Martin, W., Swinnen, J., Vos, R.: Covid-19 risks to global food security. Science **369**(6503), 500–502 (2020)
6. Toyama, K.: Geek heresy: Rescuing social change from the cult of technology. PublicAffairs (2015)
7. Wolfson, M.D., Greeno, C.: Savoring surplus: effects of food rescue on recipients. Journal of Hunger & Environmental Nutrition (2018)
8. Wolpert, D.: Stacked generalization. Neural networks (1992)