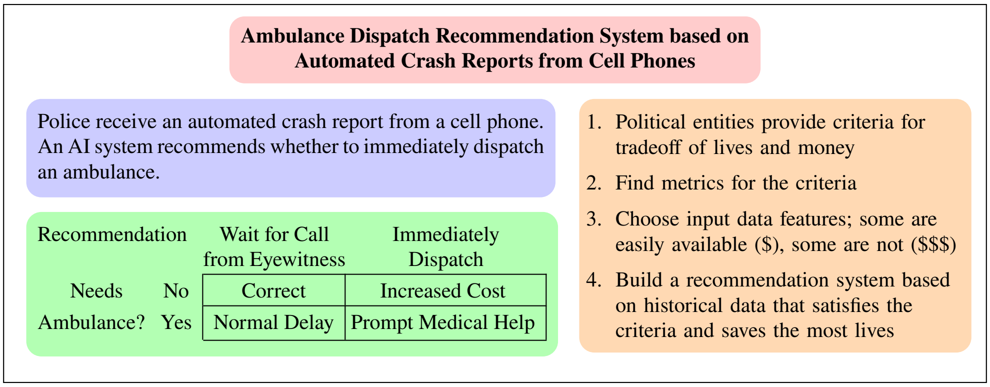
Graphical Abstract

Ambulance Dispatch Recommendation System based on Automated Crash Reports from Cell Phones

J. Bradford Burkman,Chee-Hung Henry Chu,Miao Jin,Malek Abuhijleh,Xiaoduan Sun



Highlights

Ambulance Dispatch Recommendation System based on Automated Crash Reports from Cell Phones

J. Bradford Burkman,Chee-Hung Henry Chu,Miao Jin,Malek Abuhijleh,Xiaoduan Sun

- Supports transferability and benchmarking of different approaches on a public large-scale dataset. We have attached the code we used to perform the analysis on data from the Crash Report Sampling System (CRSS).
- Novel Application motivated by Emerging Technology: Machine Learning Classification Models for Dispatching Ambulances based on Automated Crash Reports
- New Use of Dataset: Used Crash Report Sampling System (CRSS), which has imputed missing values for some features, but not all of the ones we wanted to use. For the first time we have seen, we used the software the CRSS authors use for multiple imputation (IVEware) to impute missing values in more features, then compared the results with other imputation methods.
- Explicit Incorporation of Imbalanced Costs
- Explicit Incorporation of Political Dimensions
- Consideration of Marginal Effects of Threshold Shifting
- Perennial Machine Learning Challenge: Imbalanced Datasets

Ambulance Dispatch Recommendation System based on Automated Crash Reports from Cell Phones

J. Bradford Burkman[a,b,*,1], Chee-Hung Henry Chu[a], Miao Jin[a], Malek Abuhijleh[a,c] and Xiaoduan Sun[c]

[a]*School of Computing and Informatics, University of Louisiana at Lafayette, 301 E. Lewis St, Lafayette LA, 70503, USA*
[b]*Louisiana School for Math, Science, and the Arts, 715 University Pkwy, Natchitoches LA, 71457, USA*
[c]*Department of Civil Engineering, University of Louisiana at Lafayette, 131 Rex St, Lafayette LA, 70504, USA*

ARTICLE INFO

Keywords:
Automated crash notification
Ambulance dispatch
Emergency medical services
Machine learning
Imbalanced Cost
Imbalanced Data
Imputation

ABSTRACT

Some new cell phones can automatically notify an emergency dispatcher if the phone detects the deceleration profile of a vehicular crash. Most crash notifications come from an eyewitness who can say whether an ambulance is needed, but the automated notification from the cell phone cannot provide that information directly. Should the dispatcher immediately send an ambulance before receiving an eyewitness report? There are three options: Always, Wait, and Sometimes. The "Always" option refers to sending an ambulance to every automatically reported crash, even though most of them will not be needed. In the "Wait" option, the dispatcher sends police, but always waits for a call from an eyewitness (perhaps the police) before sending an ambulance. In the "Sometimes" option, the dispatcher relies on a machine learning recommendation system to decide whether to immediately dispatch an ambulance, reserving the option to send one later based on an eyewitness report.

This paper explores one option for building a machine learning (ML) model for making a recommendation in the "Sometimes" option. Our goal is to build a model that returns, for each feature vector (crash report, sample), a value $p \in [0, 1]$ that increases with the probability that the person needs an ambulance. Then we choose a threshold θ such that we immediately send ambulances to those automated crash reports with $p > \theta$, and wait for eyewitness confirmation for those reports with $p < \theta$. In an actual implementation, the choice of θ is political, not technical, so we consider and interpret several options.

Once a threshold has been chosen, the costs of the false positives (FP) and false negatives (FN) in dispatching ambulances are very different. The cost of sending an ambulance when one is not needed (FP) is measured in dollars, but the cost of not promptly sending an ambulance when one is needed (FN) is measured in lives. Choosing the decision threshold θ is ethically problematic, but governments implicitly choose such a tradeoff when they set budgets for emergency services.

We consider and interpret several options for the decision threshold θ based on the political consideration, "How much will it cost?" How many automated ambulance dispatches are we willing to fund (FP + TP) for each one of them that is actually needed (TP)? We will explore two versions of that question, the total and the marginal.

We show that the quality of the model depends highly on the input data available, and we considered three levels of data availability. The "Easy" level includes data the emergency dispatcher has before the notification, like time of day and weather. The "Medium" level adds information about the location and information from the cell service provider about the user, like the age and sex. The "Hard" level adds information that requires having access to records about the vehicle likely to be driven by the cell phone user and detailed and temporal information about the location, like lighting conditions and whether it is currently a work zone.

We used the data of the Crash Report Sampling System (CRSS) to validate our approach. We have applied new methods (for this dataset in the literature) to handle missing data, and we have investigated several methods for handling the data imbalance. To promote discussion and future research, we have included all of the code we used in our analysis.

Contents

`bradburkman@gmail.com` (J.B. Burkman)
`http://www.github.com/bburkman/Ambulance_Dispatch` (J.B. Burkman)
ORCID(s):

List of Figures

List of Tables

1. Introduction

1.1. Overview

[Write overview]

To train the models we would wish to have actual data on crashes with automated crash reports from cell phones, but no large public dataset exists. As a proxy, we used the 2016-2021 data from the Crash Report Sampling System. In §2.3 we describe the dataset and how we binned features and imputed missing data.

We will also consider the cost of the data inputs. The emergency dispatcher has some information before the notification comes in, like day of week, time of day, and weather. Some data features would require significant budgets to obtain and maintain. There are other features useful in predicting whether the crash person needs an ambulance that, to have the data instantaneously available to emergency dispatchers might pose privacy and data security concerns. We will call these three categories of data features Easy, Medium, and Hard, but could also describe them as Free, Expensive, and Problematic. We discuss the data features in §2.4.

We used eight supervised learning algorithms, some with class weights and focal loss, to give 13 different models. (See §2.5.1) For each of the three political criteria we find the best model based on how many ambulances it correctly recommends for immediate dispatch within the limits of the criterion. We do this for the three sets of data features to show whether increasing the budget for data acquisition improves the system sufficiently to justify the cost, understanding that “sufficiently” is a political, not technical, decision.

1.2. Scenario

In the (fictitious) city of Springfield, the city council and mayor are debating whether to immediately dispatch ambulances based on automated notifications from cell phones. Many residents have cell phones (iPhones and Google Pixels) whose accelerometers will detect the deceleration profile of a crash and automatically notify the emergency call center, which immediately dispatches a police officer. The government officials are pleased that, because of the automated notifications, the police response to the crash scene is faster. Should they also immediately dispatch an ambulance, making the medical response faster?

Traditionally, the emergency call center did not know about a crash until an eyewitness called, and the eyewitness could say whether the crash persons needed an ambulance, but that information does not come with an automated crash notification from a cell phone. The notification will come with a location, the emergency dispatcher already has some information (time of day, day of week, weather, urbanicity), and the cell service provider may provide some information about the primary user of the cell phone (age, sex). With that information, the emergency dispatcher has three options.

- Always immediately dispatch an ambulance, most of which will not be needed
- Never immediately dispatch an ambulance; instead, wait for a call from an eyewitness. Many of the ambulances eventually sent to crashes had a cell phone notification and could have been sent sooner.
- Sometimes. Develop and implement an AI recommendation system to decide which to send immediately, reserving the option to send an ambulance later based on a call from an eyewitness.

In Springfield today, without immediate ambulance dispatch based on automated crash notifications from cell phones, 50% of dispatched ambulances go to automobile crashes and 10% of crash persons need an ambulance. Twenty percent of the crashes first have an automated notification from a cell phone, then a call from an eyewitness telling whether or not the crash person needs an ambulance. The other 80% of crashes only have an eyewitness call. Of the crashes with automated notifications from cell phones, 15% will need an ambulance, and 85% will not. In Figure 1 we have scaled the numbers per 100 ambulances sent before implementation of immediate ambulance dispatch.

(We chose these numbers for clarity of explanation, and an actual implementation would use local data. For details on the 85/15 split, see §2.3 Dataset and §3 Simplifying Assumptions.)

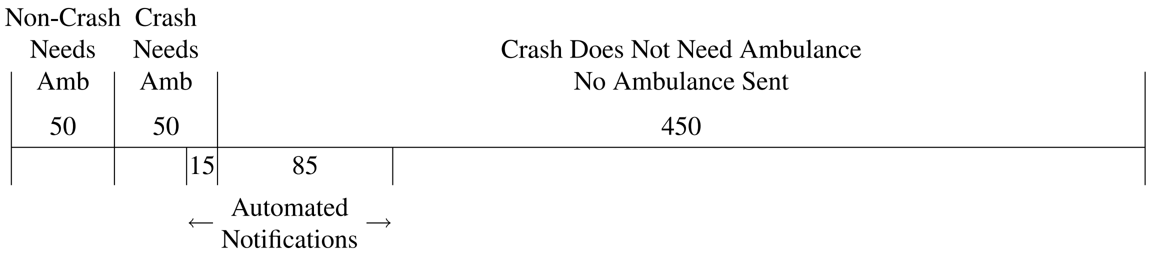


Figure 1: Springfield before implementing immediate dispatch of ambulances. Figure accompanies §1.2

If Springfield were to implement an AI recommendation system to immediately dispatch ambulances based on automated calls from cell phones, the recommendations would not perfectly predict which crash persons need an ambulance. See Figure 2, where we have zoomed in on the left side of Figure 1. In our per-100-ambulances-currently-sent proportions, the recommendation system would classify each of the automated notifications as needing or not needing an ambulance.

Of the fifteen automated crash notifications that need an ambulance, the system would correctly classify some of them as needing an ambulance (True Positives, TP), and those crash persons would get medical attention more promptly, which is the goal and benefit of the recommendation system. The rest of those fifteen would be incorrectly classified as probably not needing an ambulance with a recommendation to wait for a call from an eyewitness before sending one. (False Negatives, FN). Note that the false negatives get an ambulance just as quickly under the new system as under the old, with an ambulance dispatched upon call from an eyewitness.

Of the 85 automated notifications that do need an ambulance, some would be correctly classified (True Negatives, TN), but some would be incorrectly classified and we would immediately dispatch an unneeded ambulance (False Positives, FP). Besides administration, those additional ambulance runs are the cost of immediately dispatching ambulances. In the short term those additional ambulance runs could be more than current resources (ambulances and their teams) could handle, and in the long term could be unacceptably expensive.

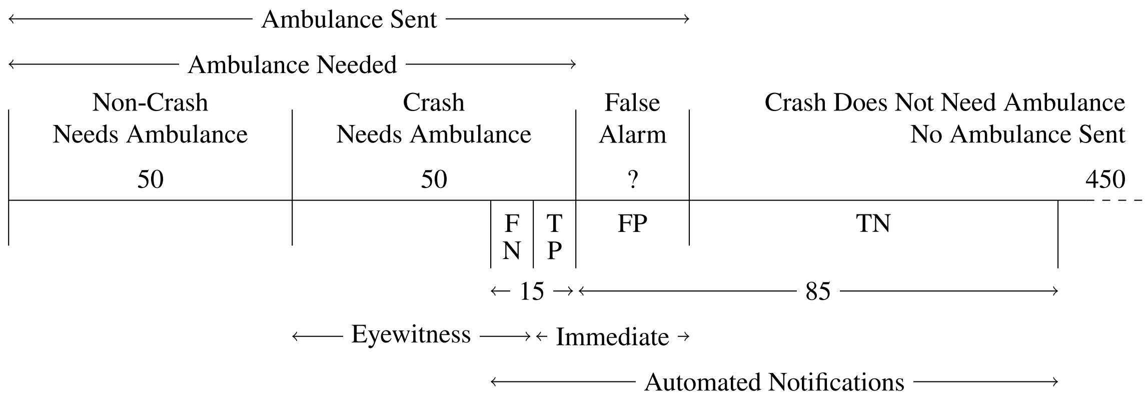


Figure 2: Springfield after implementing immediate dispatch of ambulances. Figure accompanies §1.2

The leaders of Springfield need to find a balance between the benefit of more prompt medical attention and the cost of sending more ambulances. The tradeoff of lives and money is not ethically or morally comfortable, but that is the choice governments make when they set budgets for health care and emergency services. In the confusion matrices in Figure 3, Springfield would love to increase TP without increasing FP, but the recommendation system will not give perfect predictions.

		Prediction	
		PN	PP
Actual	N	TN	FP
	P	FN	TP

Recommendation		Wait for Call from Eyewitness	Immediately Dispatch
Needs	No	Correct	Increased Cost
Ambulance?	Yes	Normal Delay	Prompt Medical Help

Figure 3: Confusion matrix for ambulance dispatch. Figure accompanies §1.2

Building Springfield's AI recommendation system starts with an historical dataset with the features the emergency dispatchers will have at the time of the automated notification, like time of day, weather, maybe age and sex, and possibly more information, and whether that historical crash person needed an ambulance (supervised learning). A machine learning algorithm learns a model of the data, and when an automated crash notification comes in, given the data available, the model returns a value $p \in [0, 1]$ that increases with the probability that the crash person needs an ambulance. Choosing $p = 1$ would mean never immediately dispatching an ambulance, and $p = 0$ would be always. The city council and mayor need to choose a decision threshold θ such that if, for a particular crash notification, $p > \theta$, then immediately dispatch an ambulance; if $p < \theta$, wait for a call from an eyewitness.

The histogram in Figure 4 shows typical model output. The model generally gives lower p values to crash persons who do not need an ambulance (Neg) and higher p values to crash persons who do need an ambulance (Pos), but there is significant overlap. The most obvious feature of the histogram is the class imbalance, that there are many more Neg than Pos, in fact $85/15 \approx 6$ Neg for each Pos.

Given a choice of θ, Springfield would immediately dispatch ambulances to all of the crashes to the right of θ. The Pos (Needs ambulance) to the right of θ (TP) would get more prompt medical attention, but the Neg (Does not need ambulance) to the right (FP) would be wasted ambulance runs. At $\theta = 0.8$, TP and FP are about equal, but as we consider smaller θ the number of TP increases by smaller and smaller amounts while the number of FP grows dramatically.

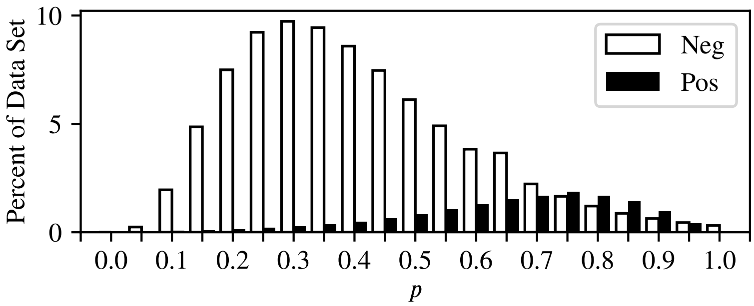


Figure 4: Example model test results. Figure accompanies §1.2

We will consider three ways the leaders of Springfield can think about how to choose θ, three metrics for political decision thresholds, detailed in §2.2.

1. Percent increase in number of ambulance calls
2. Percent of immediately dispatched ambulances that are actually needed
3. Minimum probability that an immediately dispatched ambulance is actually needed

2. Methods

2.1. Outline

1. Political Goals

2. Metrics for Political Goals
3. The Dataset
4. Choosing Features
5. Building Models
6. Analysis of Cost/Benefit
7. Choosing the Best Model

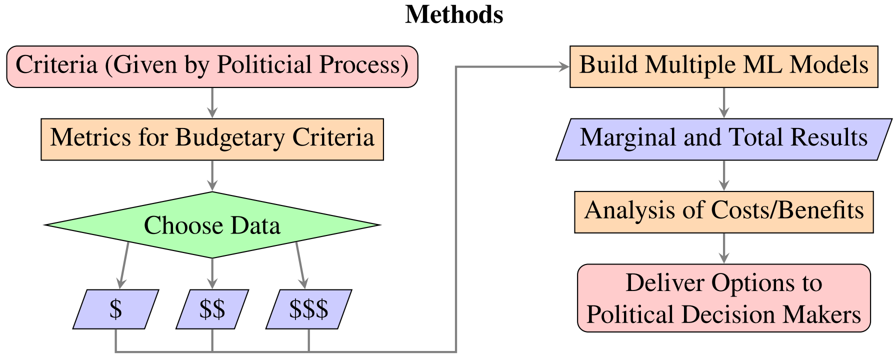


Figure 5: Methods Graphical Abstract

2.2. Budgetary Decision Thresholds and Corresponding Metrics

Saying that we trade off lives for money makes us uncomfortable, but that's what governments do when they set budgets for health care and emergency services. Our budgets are finite, and spending more money has diminishing returns, so we have to choose some criteria for our decision and accompanying metrics that let us quantify the criteria and choose an appropriate decision boundary for our recommendation system.

In our Springfield scenario,

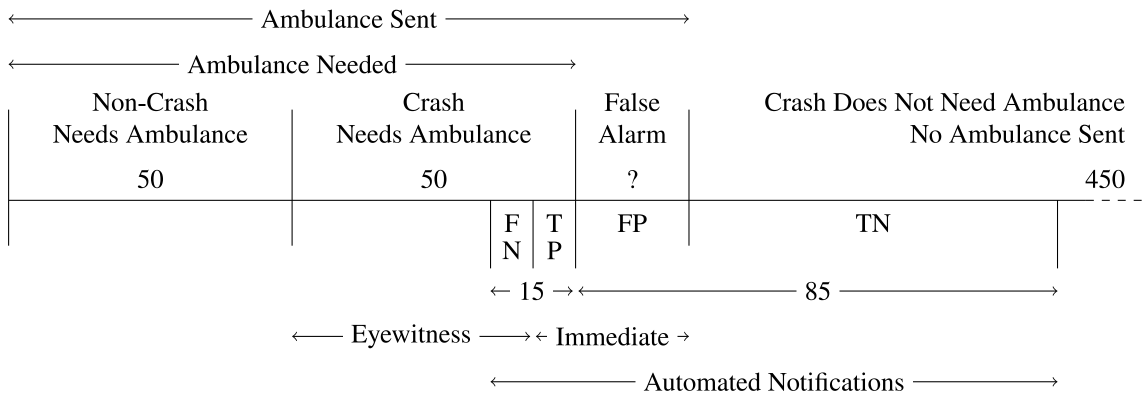


Figure 6: Springfield after implementing immediate dispatch of ambulances. Figure accompanies §1.2

2.2.1. Budgetary Decision Metric I: Percent Increased Number of Ambulance Calls

When a city or region implements immediate dispatch of ambulances, the number of ambulance dispatches increases by FP. Within the automated notifications, P = FN + TP ambulance runs becomes P + FP = FN + TP + FP ambulance runs, an increase by a rate of FP/P. The increase does not include the true positives (TP), because those ambulances would go eventually with or without immediate dispatch; the increase is the number of false positives (FP).

In the short term (too short to buy more ambulances and hire more teams), the existing budget can support an increase of the number of ambulance runs to crash persons by some small percentage. In the longer term, the city is willing to increase the budget to increase the number of ambulances going to crashes with automated notifications by a larger, but still fixed, percentage. We will use 5% as our example of how to implement this policy.

In our Springfield scenario, scaling to 100 currently sent ambulances to crash or non-crash, the total number of ambulance runs goes from 100 to 100 + FP, a rate of increase of FP/100. For any city or region, if we knew the proportion of crashes with automated notifications from cell phones and the proportion of ambulances going to crashes, we could choose an FP/P threshold to match a budgetary decision criterion based on the increase in total number of ambulances sent to crashes or total number of ambulances sent to any situation.

Set the decision threshold θ where the number of false positives is 5% of the positive class.

$$\frac{\mathrm{FP}}{\mathrm{P}} = \frac{\mathrm{FP}}{\mathrm{FN} + \mathrm{TP}} = 0.05 \tag{1}$$

2.2.2. Budgetary Decision Metric II: Percent of Immediately Dispatched Ambulances Actually Needed

A city or region is willing to immediately dispatch ambulances based on automated crash reports, but only up to the point where a certain proportion of the ambulances they immediately dispatch (PP = FP + TP) are actually needed (TP). This proportion, TP/(FP + TP) is called the *precision* of a machine learning model.

To illustrate the method, we will choose Precision = 2/3, being willing to immediately dispatch one unnecessary ambulance for each two necessary ones.

$$\frac{\mathrm{TP}}{\mathrm{PP}} = \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP}} = \frac{2}{3} \tag{2}$$

2.2.3. Budgetary Decision Metric III: Minimum Probability that Each Immediately Dispatched Ambulance is Needed

The previous two decision criteria let the city leaders choose a specific dollar amount of increase in the annual ambulance budget, but for ethical reasons they may decide that, while they cannot afford to immediately dispatch an ambulance to every crash notification, they should immediately dispatch an ambulance to a crash notification with some probability (like 50% or 80%) of needing medical attention, and consider the total cost later.

Our recommendation system will use a supervised-learning binary classification model trained on historical data. The models do not actually return a probability for each sample. The models return, for each sample, a value p that, generally, increases with the probability. For each sample we also know whether that historical crash person actually needed an ambulance, whether that sample is in the negative or positive class.

Consider a small band of values $p \in [\theta, \theta + \delta)$. The samples in that band of p are either in the negative or positive class. Call the number of negative and positive samples in the band Neg and Pos, to distinguish from the total number of samples in the negative and positive classes, N and P. The probability that a crash person in that band of p needs an ambulance is given by Pos/(Neg + Pos).

Since we have a discrete data set, not a smooth continuous function, if δ is too small, we will see the underlying randomness and the probability we calculate will not be an increasing function of p. Larger values of δ would smooth out the randomness but give us less fine control. The ideal δ is the smallest value that makes the probability increase with p. Since some of our models give p rounded to two decimal places, and because it is sufficiently small to illustrate the method, we will generally choose $\delta = 0.01$.

To illustrate the method, we will choose the minimum marginal probability to be 50%, meaning that each ambulance we immediately dispatch has at least a fifty percent chance of being needed. We will find the least value of θ such that in the band $p \in [\theta - \delta, \theta + \delta)$,

$$\frac{\text{Pos}}{\text{Neg} + \text{Pos}} \geq 0.5 \tag{3}$$

and recommend immediately dispatching an ambulance to each automatic notification for which our model returns $p \geq \theta$.

We can relate this metric to a familiar metric if we rephrase the probability as the ratio of needed to unneeded ambulances immediately dispatched. For instance, a 50% probability is a 1:1 ratio of needed to unneeded, and an 80% probability is a 4:1 ratio of needed to unneeded. The proportion of needed to unneeded ambulances sent in a neighborhood of p is proportional to a widely used metric, the slope of the ROC curve, with the constant of proportionality being the class ratio.

$$\frac{\text{Pos}}{\text{Neg}} = \frac{\Delta\text{TP}}{\Delta\text{FP}} = \frac{\text{P}}{\text{N}} \cdot \frac{\Delta\text{TP}/\text{P}}{\Delta\text{FP}/\text{N}} = \frac{\text{P}}{\text{N}} \cdot \frac{\Delta\text{TPR}}{\Delta\text{FPR}} = \frac{\text{P}}{\text{N}} \cdot m\text{ROC} \tag{4}$$

2.3. Dataset

Ideally, we would use a dataset of crashes that spawned an automated notification, but we have not found such a dataset that is publicly available. Working with such a private dataset would be an important avenue of future research. (See §3 for a list of simplifying assumptions and opportunities for future research.)

We will use the Crash Report Sampling System (CRSS) data from 2016 to 2021. The CRSS is a curated sample of crashes in the US, weighted to more serious crashes such that 17% of the crash persons needed an ambulance, significantly more than the proportion of all reported crashes needing an ambulance. Since many low-speed crashes would have a crash profile similar to hard braking, they would not spawn an automated notification, so it is reasonable to assume that the set of crashes with automated notifications would have a higher percentage of persons needing an ambulance.

We make some simplifying assumptions (see §3) using this dataset, including that the class ratio (P:N) in the automated crash notification from cell phones will be close to that in the CRSS data, 1:5, and that the crash persons in the CRSS data are representative of the future crash persons whose cell phones send a crash notification.

We will use the CRSS as a proxy for the set of crashes with automatic crash notifications, acknowledging that we do not know how good of a proxy it is. The primary merit of CRSS for our work is that it is publicly available so that our work can be critiqued, adapted, and expanded by others.

To prepare the data we had to bin (discretize) some features and to impute missing data. Some features in CRSS have both the original data with values signifying "Missing" or "Unknown" and a new feature with missing values imputed using IVEware (Raghunathan, Solenberger, Berglund and van Hoewyk), but not all of the features we wanted to use had imputed values (Herbert, 2019), so we compared several methods . We debated the proper order of operations for binning and imputing, tried both, and decided to bin first, then imputed using a round-robin random forest method. We removed all crashes involving a pedestrian because deceleration profile of such a crash would be more like hard braking than hitting a large immovable object like a car or tree, so less likely to trigger an automated notification.

The dataset is slightly imbalanced, with five element of the negative class for each element of the positive class. We considered several methods to handle the imbalance, including resampling, class weights, focal loss (Lin, Goyal, Girshick, He and Dollár, 2017), and balanced metrics. We cannot use the popular SMOTE oversampling method because our data is categorical (Chawla, Bowyer, Hall and Kegelmeyer, 2002). We tried undersampling with Tomek Links, but the resulting model results were not significantly different. The best results came from using the model algorithms from Imbalanced-Learn (Lemaître, Nogueira and Aridas, 2017), some of which apply bagging on top of algorithms from Scikit-Learn (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay, 2011).

After removing those pedestrian crashes we had 713,566 samples, each representing a crash person, and 78 relevant features. For details, see these sections of code at `www.github.com/bburkman/Ambulance_Dispatch`.

```
Ambulance_Dispatch_01_Get_Data.ipynb
Ambulance_Dispatch_02_Correlation.ipynb
Ambulance_Dispatch_03_Bin_Data.ipynb
Ambulance_Dispatch_04_Impute_Missing_Data.ipynb
Ambulance_Dispatch_05_IVEware_Order_of_Operations.ipynb
Ambulance_Dispatch_06_Build_Models_Tomek_Links.ipynb
```

2.4. Choosing Features

The `Accident`, `Vehicle`, and `Person` files of the CRSS dataset 2016-2021 have 170 unique features.

First we want to narrow the features to those that are relevant, of good quality, and knowable at the at the time of the automated notification (before any eyewitness reports). Some features, like Vehicle Identification Number (VIN), have no predictive value. Other features have missing data for more than 20% of samples. Some features, like drug and alcohol test results, are unknowable at the time of the automated notification.

Having data available for instantaneous analysis when the crash notification comes in is not free, and some features are more expensive than others. A city thinking of implementing a recommendation system for immediate dispatch will need to decide how much to spend to have the data available, and whether the more expensive features increase the quality of the models enough to be worth the cost. We categorized the features as “Easy,” “Medium,” and “Hard,” which can also be called “Free,” “Expensive,” and “Problematic.” The “Easy” features are those the dispatcher already has, like day of week, time of day, weather, and urban/rural. The “Medium” features add details about the location (intersection, speed limit, interstate highway) and information the cell service company probably has about the primary user of the phone (age and sex). To have the medium features instantaneously available would require coordination of many resources and be expensive to set up. The “Hard” features are much more problematic, requiring more coordination of public and private records, and introduce privacy and data security issues. Hard features include whether the location is a work zone, the likely vehicle driven by the primary user of the cell phone, and, if there are multiple automated notifications from the same location, how many crash persons are likely to be involved.

We note here our simplifying assumption (§3) that we will have complete and accurate data for each automated notification. Also, we will test three combinations of features but have not done more detailed testing to see which individual features or groups of features are most or least useful in predicting whether a crash person needs an ambulance.

See `Ambulance_Dispatch_01_Get_Data.ipynb` for a list of the excluded features.

See `Ambulance_Dispatch_03_Bin_Data.ipynb` for a list of the features we used for imputation of missing data in CRSS.

See `Ambulance_Dispatch_07_Build_Models.ipynb`. for the complete list of the features used in the Easy, Medium, and Hard model building.

2.5. Models

See `Ambulance_Dispatch_07_Build_Models.ipynb` for more details.

2.5.1. Binary Classification Algorithms and Hyperparameters

For each of the three sets of features we used eight binary classification algorithms, three of which take class weights α and one of which takes the focal loss parameter γ. (See Table 2.5.1) We built models for various values of the hyperparameters, giving $3 \times 13 = 39$ different models. The $\alpha = 0.5$ class weight is the default, and the $\alpha = 0.85$ class weight balances the effect of the negative and positive class in the loss function, as 85% of the samples are in the negative class. Focal loss (Lin et al., 2017) puts more weight in the loss function on the samples that are badly classified, much like least squares regression puts more weight on the points furthest from the line. Setting $\gamma = 0.0$ has no effect; Lin's paper tested from $\gamma = 0.5$ to $\gamma = 5.0$ and recommended $\gamma = 2.0$.

Table 1
Models Tested for Recommendation System. Table accompanies §2.5.1

Model	**Source**	**Class Weight α**	**Focal Loss γ**
AdaBoost Classifier	Scikit-Learn		
Balanced Bagging Classifier	Imbalanced-Learn		
Balanced Random Forest Classifier	Imbalanced-Learn	0.5	
		0.85	
Easy Ensemble Classifier with AdaBoost Estimator	Imbalanced-Learn		
KerasClassifier with the	Keras	0.5	0.0
Binary Focal Crossentropy loss function		0.5	1.0
		0.5	2.0
		0.85	0.0
Logistic Regression Classifier	Scikit-Learn	0.5	
		0.85	
Random Forest Classifier	Scikit-Learn		
RUSBoost Classifier	Imbalanced-Learn		

2.5.2. Five-Fold Cross Validation

As mentioned in §2.2.3, we need enough samples in each band $p \in [\theta, \theta + \delta)$ to smooth out the randomness. With more samples, we can use a smaller δ and be more precise in our specification of the decision threshold θ. If we used a typical 70/30 train/test split, we would only have p values for the 30% samples in the test set. Instead we used five-fold cross validation, having an 80/20 split five times, giving us p values for all of the samples. With about seven hundred thousand samples, choosing $\delta = 0.01$ for 100 bands of $p \in [0, 1]$, we have an average of seven thousand samples in each band.

2.5.3. Interpreting Supervised Learning Binary Classification Results

In supervised learning binary classification, a model predicts, for each sample in the test set, whether the sample is in the negative or positive class. The model returns a value $p \in [0, 1]$, that increases with the probability that the sample is in the positive class. Additionally in supervised learning, we already know the answer to the question of whether the sample was in the negative or positive class, with $y \in \{0, 1\}$ given in the dataset but hidden from the model during the test phase. In the code, p is often called `y_proba` and y is called `y_test`. Using these two numbers, p and y, we can study, quantify, and illustrate how well the model predicts the actual values.

A perfect model would entirely separate the negative and positive classes, but the ideal we can hope for is that most of the negative elements are towards the left and most of the positive elements are towards the right of the distribution. In Figure 7, the data has the same class ratio as our CRSS data, with 85% in the negative class and 15% in the positive class. If we choose discrimination threshold $\theta = 0.5$, the value of p for which most models algorithms are optimized, the elements of the negative class with $p < \theta$ are True Negatives (TN), the elements of the negative class with $p > \theta$ are False Positives (FP), the elements of the positive class with $p < \theta$ are False Negatives (FN), and the elements of the positive class with $p > \theta$ are True Positives (TP).

If this ideal model were our recommendation system with $\theta = 0.5$, then 38.5% of the ambulances we immediately dispatched would be needed (Precision), and 73% of the needed ambulances would be immediately dispatched (Recall). If we chose a higher value of θ, we would increase TN, decrease FP, increase FN, and decrease TP. Recall would decrease, but the effect on precision is uncertain as FP and TP would both decrease.

The ROC (Receiver Operating Characteristic) curve is a parameterized curve showing the True Positive Rate (TP/P) versus the False Positive Rate (FP/N) as p varies from 0 (upper right) to 1 (lower left). The area under the curve (AUC) is widely used to compare the quality of models in terms of how well they separate the negative and positive classes over the entire range of $p \in [0, 1]$. For our work, however, given real-life budgetary constraints on expanding ambulance fleets, we are only interested in a small range of p on the right side of the distribution, so the ROC AUC is not the primary measure we will use to choose the best model.

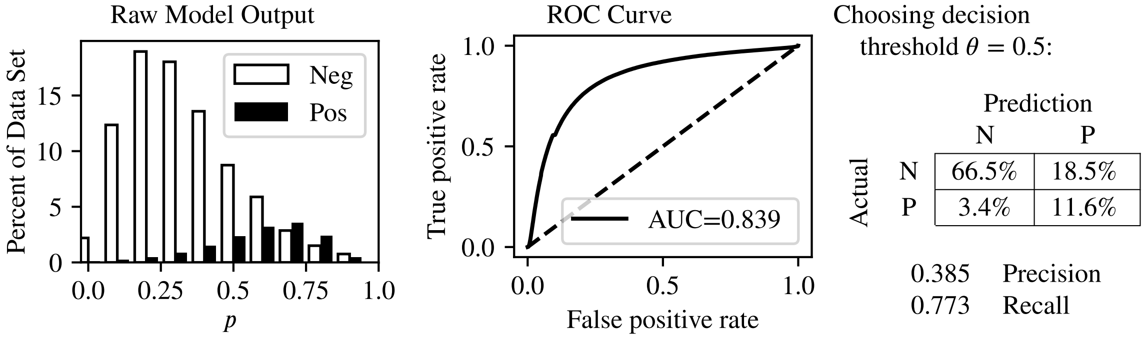


Figure 7: Example Model Results. Figure accompanies §2.5.3

2.6. Comparing Outputs of Different Models

2.6.1. Raw Model Outputs

The eight models not only give different results, but different kinds of results, and we have to find a way to compare them. See Figure 8. For the illustrations we have used models built on the Hard features with no class balance nor focal loss.

The ranges and shapes of the distributions are significantly different. The Balanced Bagging and Balanced Random Forest classifiers gives a nice spread from 0.0 to 1.0, but the AdaBoost, Easy Ensemble, and RUSBoost results are clustered in the middle, and the Random Forest on the left. If we used the Random Forest results with $\theta = 0.5$, we would not immediately dispatch any ambulances. The KerasClassifier and Logistic Regression Classifier tend towards the left with long tails to the right.

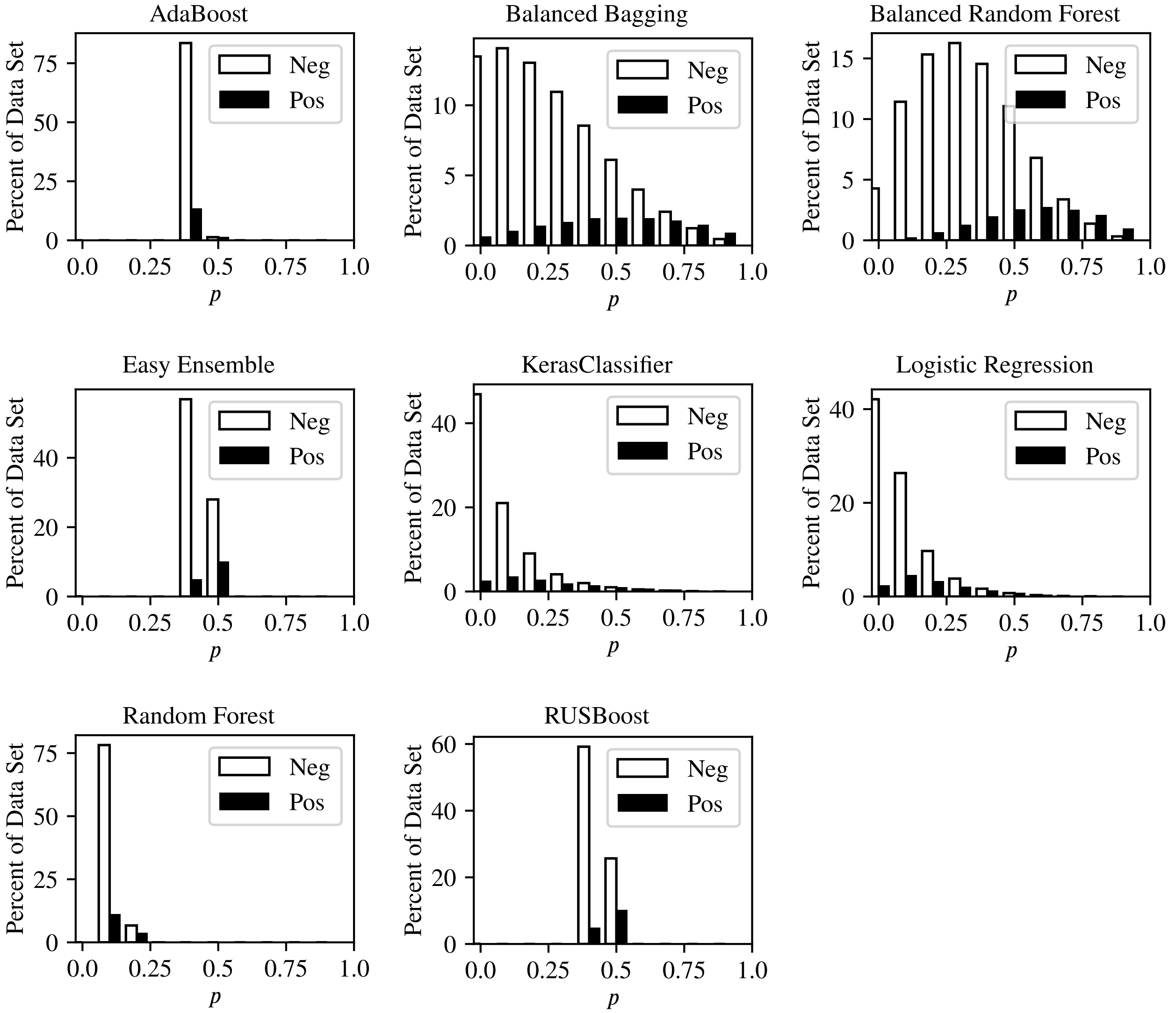


Figure 8: Raw Model Outputs. Figure accompanies §2.6.1

2.6.2. Numerics

We also need to be careful with the numerics, because the results could depend on how we slice p into δ-intervals. Table 2 shows, for each of the eight classifiers with the hard features and no class weights nor focal loss, the number of samples (always 713,566), the number of unique values of p, the sum of the value counts of the ten (and hundred) most common values of p, the min and max of p, and the area under the ROC curve.

The p distribution from RUSBoost only ranges from 0.4990 to 0.5011, but within that 0.0021 range, the 713,566 samples have 706,938 unique values of p, which is as close to "continuous" as we can hope. On the other extreme, the Balanced Bagging distribution has only 270 unique values of p, and the ten most common values comprise 99% of the set, making it very discrete. Almost all of the values of p for Balanced Bagging are rounded to one decimal placeand 93% of the p values from Balanced Random Forest are rounded to two decimal places, which is important to acknowledge because we cannot claim to find a best value of θ with more precision than the outputs of the model.

The table gives the area under the ROC curve, a common metric for comparing models in terms of how well they separate the positive and negative classes over the entire interval $p \in [0, 1]$. All of the models in this table are "good,"

Table 2
Numerics of Model Outputs of p. Table accompanies §2.6.2

	AdaBoost	BalBag	Bal RF	EasyEns	Keras	LogReg	RF	RUSBoost
n	713,566	713,566	713,566	713,566	713,566	713,566	713,566	713,566
p unique	705,474	270	3,999	3,015	694,949	706,940	458,530	706,938
Top 10	104	706,118	127,761	67,307	103	101	839	101
Top 100	404	713,239	662,937	294,986	423	392	5,872	392
min(p)	0.486	0.0	0.0	0.437	0.0	0.004	0.11	0.4990
max(p)	0.5066	1.0	1.0	0.592	0.996	0.91	0.31	0.5011
ROC AUC	0.753	0.763	0.801	0.730	0.778	0.735	0.708	0.754

with the Balanced Random Forest best and Random Forest worst, but the differences we are interested in are in a small interval of p that satisfy the budgetary decision criteria, so the AUC will not be the primary metric we use.

2.6.3. Transforming Model Outputs

There are several ways we could handle the small or left-leaning ranges of the p-output of the models, but we have chosen to linearly transform the values of p so that the transformed distributions look like the smooth and full-range graph of the Balanced Random Forest Classifier, for two reasons. First, the transformed graphs are much more effective visualizations for gaining insight about the data. Second, we want to have enough positive and negative elements in each of a hundred even intervals of [0, 1] to smooth out the randomness so that we can choose a value of θ accurate to two decimal places. (Except for the Balanced Bagging model, where we can only have ten intervals.)

To transform the p values from the RUSBoost Classifier (see Figure 9), we mapped the 1% and 99% quantiles to 0 and 1, respectively, drew a line between those two points, then mapped everything below 0 to 0 and everything above 1 to 1.

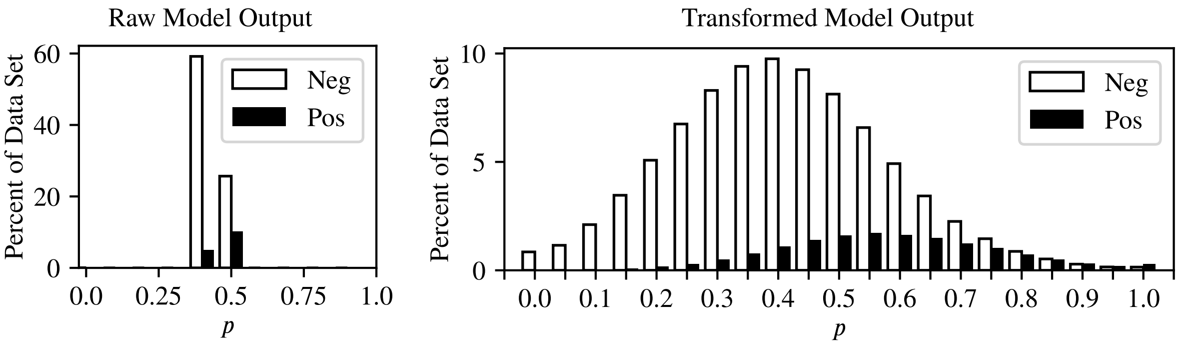


Figure 9: RUSBoost Classifier. Figure accompanies §2.6.3

The Logistics Regression data (see Figure 10) has a long tail, so we mapped the 5% and 95% quantiles to 0 and 1; note the bump on the right where we mapped the top 5% of the values to 1. Is it a problem that we have essentially discarded the top 5% of the values of p? No, not if the value of θ that fits our criteria is less than 1.00 in our transformed data. In choosing a value of the decision threshold θ, we only care how many elements of the positive and negative classes are to the left and right of θ. Whether the elements are clustered together is not relevant.

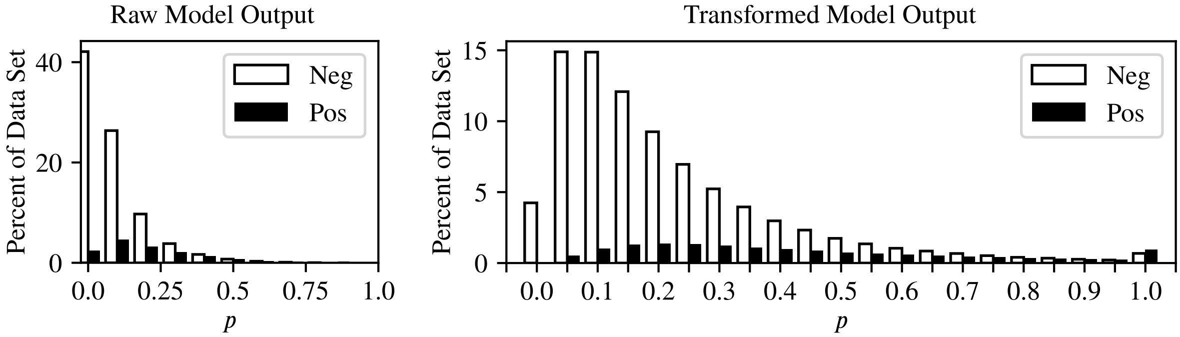


Figure 10: Transformation of the Logistic Regression Classifier Output. Figure accompanies §2.6.3

Our visualization of the transformed Random Forest Classifier output (see Figure 11) hints at trouble in using that model, that we have an actual bimodal distribution (not just because of rounding at the top), and the value of θ that satisfies each criterion may not be unique. When we wrote the code to find the optimal value of θ, we did not plan for that contingency.

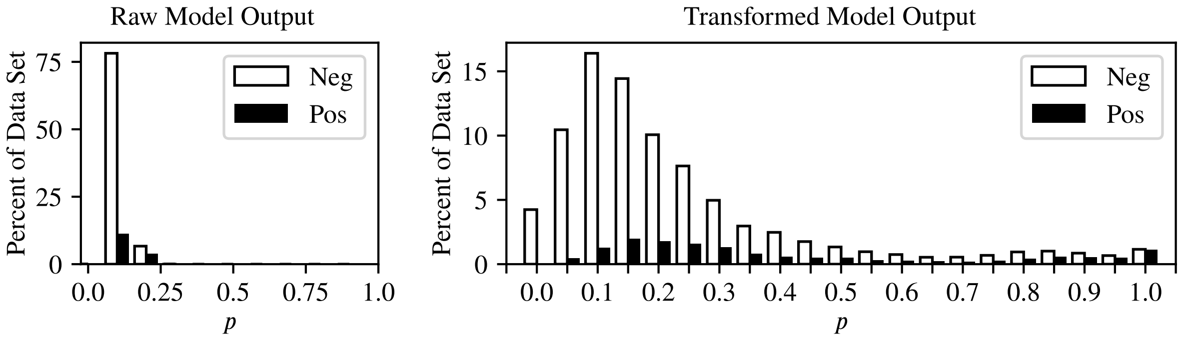


Figure 11: Transformation of the Random Forest Classifier Output. Figure accompanies §2.6.3

How we should transform the Easy Ensemble Classifier data is sensitive to the numerics. The raw p output has 3,015 unique values, which is better than the 270 from the Balanced Bagging Classifier, but not nearly as continuous as the $\approx 700,000$ unique values of four of our models. Depending on where we slice the intervals, we may not see the smooth curve that we hope underlies the results.

In Figure 12, we mapped the min and max to 0 and 1, respectively, but we want to see more samples returning values of p greater than 0.8, so in Figure 13 we rounded the tails, mapping the 1% and 99% quantiles to 0 and 1.

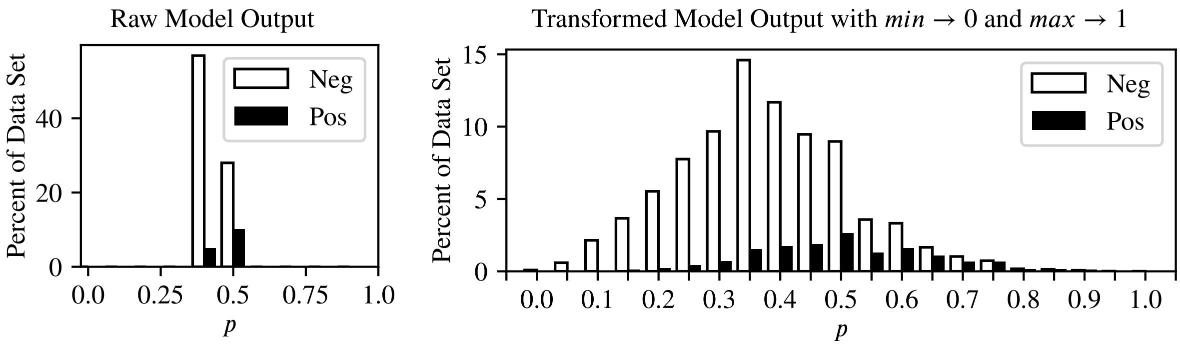


Figure 12: One Transformation of Easy Ensemble Output. Figure accompanies §2.6.3

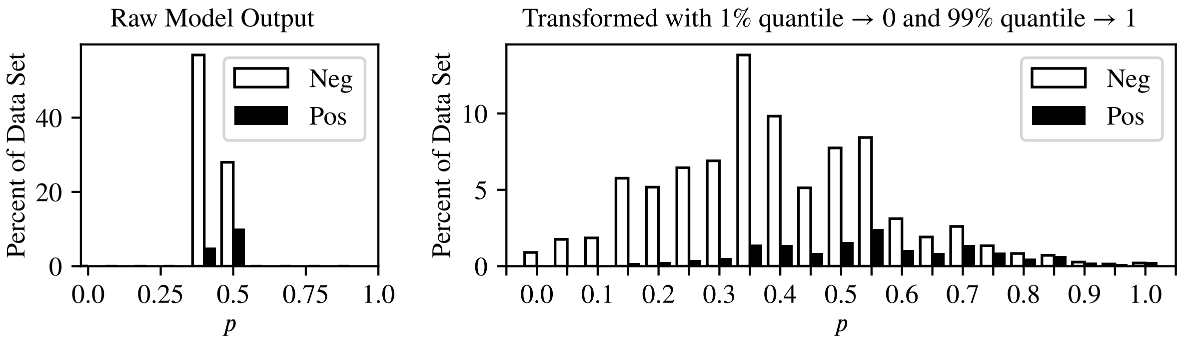


Figure 13: Another Transformation of Easy Ensemble Output. Figure accompanies §2.6.3

Except for values of p in the truncated quantiles, the transformations are invertible, so choices of $\theta < 0.99$ in the transformed data can be converted to a choice of p-threshold in the original model output.

2.6.4. Hyperparameters

Our experience with varying class weights and focal loss parameter was that they shifted the entire p distribution, both the positive and negative class, but did not do a better job of separating the positive and negative class, as measured by the area under the curve (AUC) of the receiver operating characteristic (ROC), as illustrated in Figure 14 below.

The KerasClassifier with the Binary Focal Crossentropy loss function takes both a class weight parameter α and a focal loss parameter γ. Varying these parameters gave us different shapes of distributions of p, but all three versions of the model had the ROC AUC between 0.7781 and 0.7785, a difference within the normal ranges of randomness in machine learning models. Using our techinques described in §2.6.3, we could linearly transform the output of these models to make them all look basically the same.

If one is using the default decision threshold $\theta = 0.5$, then these hyperparameters are useful for shifting the distribution to meet the threshold, but since we are taking the liberty to move the threshold, varying the hyperparameters may be of little use. Since the ROC AUC quantifies how well the algorithm separates the positive and negative classes over the whole range of p, and we are most interested in a small range of p on the right end, the weights may have some effect, a topic for future investigation.

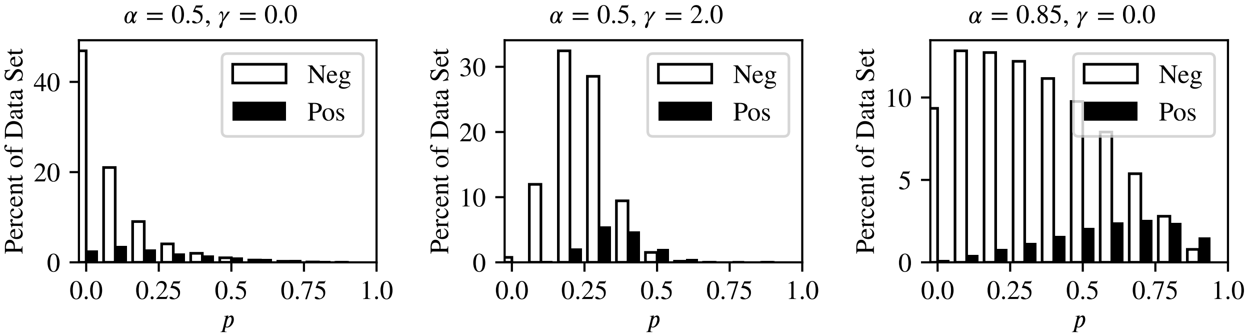


Figure 14: KerasClassifier with Different Hyperparameters. Figure accompanies §2.6.4

2.6.5. Understanding the Metrics in Bands of Values of p

In Table 3 we have various metrics as a function of p returned by the Balanced Forest Classifier. When choosing the best model for each metric we will use p-intervals of width 0.01, but for illustration purposes here we use intervals of width 0.05.

The "Neg" and "Pos" are the number of elements of each class in that interval of p. The Pos/(Neg + Pos) is one of our target metrics. The True Negatives (TN) are a running sum of Neg, and the False Positives (FP) are N − TN. Similarly, the False Negatives (FN) are a running sum of Pos, and the True Positives (TP) are P − FN. Precision, one of our target metrics is TP/(FP + TP), is the proportion of ambulances immediately dispatched that are needed. Recall, TP/(FN + TP), is the proportion of needed ambulances that are immediately dispatched. The last of our target metrics, FP/P, is the proportional increase in the number of ambulances sent (immediately or upon call from an eyewitness) when we automatically dispatch some ambulances based on an automated notification from a cell phone.

For example, if we set $\theta = 0.50$, then out of $n = 713,566$ automated crash notifications from cell phones, of the $P = 107,956$ that need an ambulance, we will send TP $= 77,763$ immediately and send the other FN $= 30,193$ after hearing from an eyewitness that an ambulance is needed. Additionally, we will send FP $= 163,691$ ambulances to crash persons who do not need one. Of the ambulances we immediately dispatched, Precision $= 32\%$ of them were needed, and of the crash persons who needed an ambulance, we immediately dispatched ambulances to Recall $= 72\%$ of them. The FP $= 163,691$ unnecessarily sent ambulances represent a FP/P $= 152\%$ increase in the number of ambulances sent to those crash persons with automated crash notifications, an increase over just ignoring the automated notifications and always waiting for a call from an eyewitness.

If we were to move from $\theta = 0.50$ to $\theta = 0.55$, then we would immediately dispatch far fewer (Neg + Pos $= 43,098 + 8,652 = 51,750$) ambulances. Pos $= 8,652$, or Pos/(Neg + Pos) $= 17\%$ of the ambulances we decided to not send because we moved from $\theta = 0.50$ to $\theta = 0.55$, were needed. In that band of θ, automated calls from cell phones have a 17% chance of needing an ambulance.

Table 3
Various Metrics as a Function of p returned by the Balanced Random Forest Classifier on the Hard Features. Table accompanies §2.6.5

p	Neg	Pos	$\frac{\text{Pos}}{\text{Neg+Pos}}$	TN	FP	FN	TP	Prec	Rec	$\frac{\text{FP}}{\text{P}}$
0.00	546	0	0.00	546	605,064	0	107,956	0.15	1.00	5.60
0.05	8,247	76	0.01	8,793	596,817	76	107,880	0.15	1.00	5.53
0.10	22,279	270	0.01	31,072	574,538	346	107,610	0.16	1.00	5.32
0.15	35,964	624	0.02	67,036	538,574	970	106,986	0.17	0.99	4.99
0.20	45,574	1,223	0.03	112,610	493,000	2,193	105,763	0.18	0.98	4.57
0.25	52,794	1,990	0.04	165,404	440,206	4,183	103,773	0.19	0.96	4.08
0.30	56,584	2,766	0.05	221,988	383,622	6,949	101,007	0.21	0.94	3.55
0.35	58,211	3,945	0.06	280,199	325,411	10,894	97,062	0.23	0.90	3.01
0.40	57,885	5,208	0.08	338,084	267,526	16,102	91,854	0.26	0.85	2.48
0.45	54,616	6,471	0.11	392,700	212,910	22,573	85,383	0.29	0.79	1.97
0.50	49,219	7,620	0.13	441,919	163,691	30,193	77,763	0.32	0.72	1.52
0.55	43,098	8,652	0.17	485,017	120,593	38,845	69,111	0.36	0.64	1.12
0.60	35,851	9,513	0.21	520,868	84,742	48,358	59,598	0.41	0.55	0.78
0.65	27,876	9,898	0.26	548,744	56,866	58,256	49,700	0.47	0.46	0.53
0.70	20,654	9,781	0.32	569,398	36,212	68,037	39,919	0.52	0.37	0.34
0.75	14,504	9,231	0.39	583,902	21,708	77,268	30,688	0.59	0.28	0.20
0.80	9,591	8,698	0.48	593,493	12,117	85,966	21,990	0.64	0.20	0.11
0.85	6,064	8,035	0.57	599,557	6,053	94,001	13,955	0.70	0.13	0.06
0.90	3,740	7,051	0.65	603,297	2,313	101,052	6,904	0.75	0.06	0.02
0.95	1,908	5,197	0.73	605,205	405	106,249	1,707	0.81	0.02	0.00
1.00	405	1,707	0.81	605,610	0	107,956	0	nan	0.00	0.00

2.6.6. Choosing Values of θ for each Budgetary Decision Metric

Using the Balanced Random Forest Classifier trained on the Hard features as an example, from the data in Table 3 we can find the decision thresholds θ that satisfy each of our three political decision criteria. In the table, $\frac{\text{FP}}{\text{P}} = 0.05$ somewhere in the interval $p \in [0.85, 0.90)$. Zooming in on that interval in Table 4, we see that if we wanted to satisfy that criterion, we would choose $\theta = 0.86$ as our decision threshold. In the table we can also see the marginal effects on FP/P of choosing a slightly larger or smaller θ instead.

Similarly, for our second political criterion Precision $= \frac{\text{TP}}{\text{FP+TP}} = \frac{2}{3}$, we would choose $\theta = 0.81$ as our decision threshold, and for marginal probability, $\frac{\text{Pos}}{\text{Neg+Pos}} = 0.50$, we would choose $\theta = 0.79$.

We cannot get more detailed values of θ for the Balanced Random Forest Classifier because almost all of the values of p in the model output are rounded to two decimal places. For all of our models, though, we would be stretching our credibility to give more precise answers because we just do not have enough data to give our criteria as monotonic functions of p over much smaller intervals of p.

Table 4
Various Metrics as a Function of p, in more detail. Table accompanies §2.6.6

p	Neg	Pos	$\frac{\text{Pos}}{\text{Neg+Pos}}$	TN	FP	FN	TP	Prec	Rec	$\frac{\text{FP}}{\text{P}}$
0.75	2,495	1,804	0.42	583,902	21,708	77,268	30,688	0.59	0.28	0.20
0.76	2,259	1,721	0.43	586,161	19,449	78,989	28,967	0.60	0.27	0.18
0.77	2,041	1,779	0.47	588,202	17,408	80,768	27,188	0.61	0.25	0.16
0.78	1,882	1,817	0.49	590,084	15,526	82,585	25,371	0.62	0.24	0.14
0.79	1,805	1,706	0.49	591,889	13,721	84,291	23,665	0.63	0.22	0.13
0.80	1,604	1,675	0.51	593,493	12,117	85,966	21,990	0.64	0.20	0.11
0.81	1,440	1,585	0.52	594,933	10,677	87,551	20,405	0.66	0.19	0.10
0.82	1,321	1,697	0.56	596,254	9,356	89,248	18,708	0.67	0.17	0.09
0.83	1,162	1,639	0.59	597,416	8,194	90,887	17,069	0.68	0.16	0.08
0.84	1,171	1,566	0.57	598,587	7,023	92,453	15,503	0.69	0.14	0.07
0.85	970	1,548	0.61	599,557	6,053	94,001	13,955	0.70	0.13	0.06
0.86	938	1,508	0.62	600,495	5,115	95,509	12,447	0.71	0.12	0.05
0.87	784	1,475	0.65	601,279	4,331	96,984	10,972	0.72	0.10	0.04
0.88	764	1,367	0.64	602,043	3,567	98,351	9,605	0.73	0.09	0.03
0.89	695	1,383	0.67	602,738	2,872	99,734	8,222	0.74	0.08	0.03
0.90	559	1,318	0.70	603,297	2,313	101,052	6,904	0.75	0.06	0.02

2.6.7. *Challenges in p-Transformations*

Table 5 illustrates the challenges. The AdaBoost raw p output are clustered in $[0.486, 0.5066]$, so the interval around $p = 0.50$ gives the value of FP/P closest to 0.05, but only because the interval around $p = 0.49$ and $p = 0.51$ give FP/P of 5.48 and 0.0, respectively. Similarly, when we trimmed the tails of the the AdaBoost p output severely, just taking the middle 80% of values, we trimmed the range where $\text{FP}/\text{P} \approx 0.05$, which we see in the table because $p = 0.99$ and $\text{FP}/\text{P} = 0.096$, not close to 0.05.

The remaining transformations of AdaBoost, 100, 98, 95, and 90, give useful results, with $\text{FP}/\text{P} \in [0.048, 0.053]$ and $\text{TP} \in [6,619, 7,097]$. Those differences are not from different model algorithms or choices of hyperparameters, but just from the numerics of different ways to slice the results into ranges with enough samples to smooth out the inherit randomness of machine learning models and illustrate the low degree of accuracy (in the general sense, not the $(\text{TN} + \text{TP})/(\text{N} + \text{P})$ sense) we can claim in model results.

While the TP-results within the different transformations of the AdaBoost results vary some, the useful AdaBoost results are clearly better than those of Easy Ensemble and clearly worse than those of the KerasClassifier with the Binary Focal Crossentropy loss function with balanced class weights and no focal loss.

Table 5
Issues in finding values of p that make FP/P closest to 0.05. Table accompanies §2.6.7

Algorithm	Features	α	γ	Trans	p	Neg	Pos	FP/P	TP	
AdaBoost	Hard			None	0.50	582,309	97,128	0.088	10,668	Discard
AdaBoost	Hard			100	0.71	1,127	1,088	0.049	6,696	
AdaBoost	Hard			98	0.83	768	735	0.053	7,097	
AdaBoost	Hard			95	0.90	619	588	0.048	6,619	
AdaBoost	Hard			90	0.97	527	520	0.049	6,747	
AdaBoost	Hard			80	0.99	779	643	0.096	11,429	Discard
Keras	Hard	0.85	0	None	0.90	1,286	1,520	0.052	10,819	
Easy Ensemble	Hard			95	0.92	2,233	2,171	0.043	4,744	

2.6.8. Choosing the Best Model for each Budgetary Decision Metric

For each budgetary constraint we want to find the model that, within the constraint, will immediately dispatch the most ambulances to crash persons who need them (TP). Using as an example our first budgetary constraint, FP/P = 0.05, we need to find, for each model, each set of hyperparameters for the model, and each transformation of the p outputs, whether there exists a neighborhood of p where FP/P is close to 0.05, then find the best θ interval in that neighborhood. Of those valid results, find the model that gives the most TP.

Table 6 shows the best results for each model algorithm. Within these, the Balanced Random Forest Classifier gives the best results, sending more needed ambulances while staying within the budgetary constraint. The KerasClassifier with the Binary Focal Crossentropy loss function is a close second, and those two are clearly better than the other six models.

Table 6
Comparing Models: Best results for each model for budgetary criterion FP/P closest to 0.05. Table accompanies §2.6.8

Algorithm	Features	α	γ	Trans	p	Neg	Pos	FP/P	TP
BRFC	Hard	0.50	0	None	0.86	938	1,508	0.047	12,447
KBFC	Hard	0.50	2.0	100	0.58	1,264	1,556	0.054	11,287
RUSBoost	Hard	0	0	100	0.71	1,245	1,200	0.054	7,336
LogReg	Hard	0.50	0	95	0.81	348	393	0.051	7,278
AdaBoost	Hard	0	0	98	0.83	768	735	0.053	7,097
BalBag	Hard	0	0	None	0.9	8,548	10,487	0.03	6,610
RFC	Hard	0	0	100	0.74	923	673	0.051	5,909
EasyEns	Hard	0	0	100	0.72	2,378	2,296	0.048	5,306

3. Simplifying Assumptions and Opportunities for Future Research

All models are simplifications, and we should acknowledge the most egregious of our simplifying assumptions. Some of the simplifying assumptions may hint at opportunities for future research.

Section	**Simplifying Assumption**
§2.2	All ambulances sent based on calls from eyewitnesses are actually needed.
	All automated crash notifications from cell phones refer to an actual crash, not just hard braking, *i.e.* the notifications have no false positives.
§2.3	The class ratio (P/N) in the automated crash notification from cell phones will be close to that in the CRSS data, $1/5$.
§2.3	The crash persons in the CRSS data are representative of the future crash persons whose cell phones send a crash notification. (Several layers to unpack here)
§2.4	The data features we seek for each automated crash notification will be available and accurate.
Section	**Opportunities for Future Research**
§2.3	Find data on crashes that spawned an automated notification from a cell phone.
§2.4	We tested only three sets of features and did not test to see which features or combinations of features were most useful in predicting needing an ambulance.
§2.6.4	We found that varying the hyperparameters for class weight and focal loss did not significantly vary the ROC AUC, a measure of how well the model separates the positive and negative classes over the whole range $p \in [0, 1]$. Do those hyperparameters make a difference in how well the model separates the positive and negative classes on the right tail of the p distribution, the part relevant to our work?

4. Results

Table 7
Best models and transformations for FP/P = 0.05 for each algorithm. Table accompanies §2.6.6

Algorithm	Features	α	γ	Trans	p	Neg	Pos	FP/P	TP
BRFC	Easy	0.50	0	98	0.93	997	564	0.052	4270
BalBag	Easy	0	0	95	0.97	755	389	0.056	4197
KBFC	Easy	0.85	0.0	None	0.75	2345	1106	0.058	3469
LogReg	Easy	0.50	0	None	0.35	1360	624	0.055	2998
EasyEns	Easy	0	0	100	0.92	1513	638	0.051	2768
AdaBoost	Easy	0	0	100	0.85	877	453	0.05	2712
RFC	Easy	0	0	100	0.89	938	452	0.051	2600
RUSBoost	Easy	0	0	98	0.96	1512	589	0.047	1849

Algorithm	Features	α	γ	Trans	p	Neg	Pos	FP/P	TP
BRFC	Medium	0.50	0	95	0.94	679	523	0.055	7290
KBFC	Medium	0.50	2.0	100	0.69	1328	1134	0.055	6595
AdaBoost	Medium	0	0	100	0.74	1114	992	0.053	5699
LogReg	Medium	0.85	0	98	0.93	1016	861	0.053	5623
EasyEns	Medium	0	0	98	0.91	812	800	0.057	5259
RFC	Medium	0	0	100	0.74	734	671	0.052	5152
BalBag	Medium	0	0	None	0.94	113	66	0.05	5120
RUSBoost	Medium	0	0	100	0.48	1702	439	0.053	2112

Algorithm	Features	α	γ	Trans	p	Neg	Pos	FP/P	TP
BRFC	Hard	0.50	0	None	0.86	938	1508	0.047	12447
KBFC	Hard	0.50	2.0	100	0.58	1264	1556	0.054	11287
RUSBoost	Hard	0	0	100	0.71	1245	1200	0.054	7336
LogReg	Hard	0.50	0	95	0.81	348	393	0.051	7278
AdaBoost	Hard	0	0	98	0.83	768	735	0.053	7097
BalBag	Hard	0	0	None	0.9	8548	10487	0.03	6610
RFC	Hard	0	0	100	0.74	923	673	0.051	5909
EasyEns	Hard	0	0	100	0.72	2378	2296	0.048	5306

5. Conclusions

6. Discussion

7. Future Work

Funding Statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

Declarations of interest: none

Acknowledgements

George Broussard contributed to this work in the NSF Research Experiences for Undergraduates program.

Data Availability

The CRSS data is publicly available at

https://www.nhtsa.gov/crash-data-systems/crash-report-sampling-system

All of the code and generated data, tables, and graphs are available at http://www.github.com/bburkman/Ambulance_Dispatch

CRediT authorship contribution statement

J. Bradford Burkman: Conceptualization, Investigation, Writing - original draft, Visualization. **Chee-Hung Henry Chu:** Supervision, Methodology, Writing - review and editing. **Miao Jin:** Supervision, Methodology. **Malek Abuhijleh:** Data curation, Investigation, Methodology. **Xiaoduan Sun:** Data curation, Writing - review and editing.

References

Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W., 2002. Smote: Synthetic minority over-sampling technique. JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH 16, 321 – 357.

Herbert, G., 2019. Crash Report Sampling System: Imputation. Technical Report DOT HS 812 795. National Highway Traffic Safety Administration.

Lemaître, G., Nogueira, F., Aridas, C.K., 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research 18, 1–5. URL: http://jmlr.org/papers/v18/16-365.html.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Raghunathan, T., Solenberger, P., Berglund, P., van Hoewyk, J., . Iveware: Imputation and variation estimation software. URL: https://www.src.isr.umich.edu/software/iveware/.