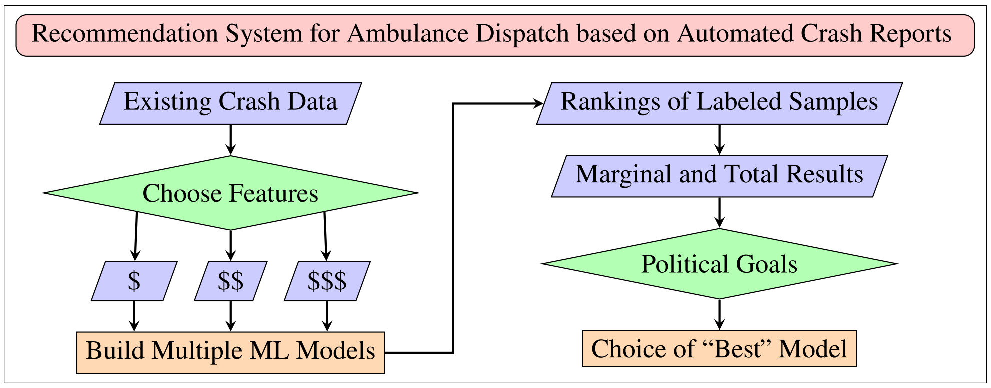
Graphical Abstract

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

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


Recommendation System for Ambulance Dispatch based on Automated Crash Reports
Existing Crash Data
Choose Features
$
$$
$$$
Build Multiple ML Models
Rankings of Labeled Samples
Marginal and Total Results
Political Goals
Choice of "Best" Model

Highlights

Recommendation System for Ambulance Dispatch 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

Recommendation System for Ambulance Dispatch 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.

1. Introduction

2. Methods

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

2.1. Overview

- The dataset
- Binning the features
- Imputing missing data
- Order of Operations in Binning and Imputing
- Handling imbalanced data
- Choosing features for “Easy,” “Medium,” and “Hard” data sets
- Building models
- Choosing metrics to interpret the results
- Interpreting the results

2.2. Metrics Old and New

In our work, we are trying to build a binary classification model to predict, based on immediately available information, whether the crash person needs an ambulance. To build a supervised-learning model, we start with labeled data, a dataset of crashes for which we know whether the person needed an ambulance. We separate the data into two sets, a training set and test set. A machine learning algorithm uses the training set to build (“learn”) a model that will return, for each crash person in a set, a value $p \in [0, 1]$ that is a proxy for the probability that the crash person needs an ambulance. We choose a decision threshold θ (by default $\theta = 0.5$) and classify samples (crash persons) with $p > \theta$ as “Needs ambulance” and samples with $p < \theta$ as “Does not need ambulance.” Then we apply the model to the test set, which the algorithm did not see in the learning phase. The test results tell us how many of the crash person who needed an ambulance or did not need an ambulance were correctly classified, and we organize them in a confusion matrix.

		Prediction	
		Predicted Negative (PN)	Predicted Positive (PP)
Actual	Negative (N)	True Negative (TN)	False Positive (FP)
	Positive (P)	False Negative (FN)	True Positive (TP)

Obviously we would prefer a model that returned no false positives or false negatives, but barring that, we would like to come as close as possible, and we need to define “close” in a measurable way.

Metric	Formula	Meaning
Accuracy	$\frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$	Proportion correctly classified
Precision	$\frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{PP}}$	Proportion of immediately-dispatched ambulances that are needed
Recall	$\frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{P}}$	Proportion of needed ambulances that are immediately dispatched
F1	$\frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$	Harmonic mean of Precision and Recall

Accuracy and precision depend on the class balance, but recall does not. Most reported crashes are fender benders that do not require an ambulance, so in our dataset, $P \ll N$. If we artificially balance the dataset so that $P = N$, either by resampling the data or using class weights in the loss function, then we would change the proportions in accuracy and precision. In the literature there is a "balanced accuracy" that is widely used. One could make a "balanced precision" in the same way, which would lead to a "balanced F1," but we have not seen those used in the literature. Recall is not affected by class balance because all of its terms (TP and FN) are in the positive class.

In combining precision and recall to make F1, why do we choose the harmonic mean instead of the arithmetic or geometric mean? There is a Gmean in the literature, but it is not used as often. If a and b are such that $0 < a < b$, then

$$a < \text{Harm}(a, b) < \text{Geo}(a, b) < \text{Arith}(a, b) < b$$

The harmonic mean leans towards the worse of the two metrics (precision and recall), much like least squares regression emphasizes the points furthest from the line, telling us not only their central tendency but also whether the two metrics are in balance.

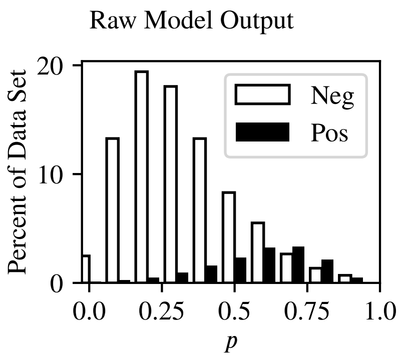
The predicted p values, when paired with the actual $N = 0$ and $P = 1$ values, combine to make the False Positive Rate and True Positive Rate values that become the parameterized curve called (for historical reasons) the Receiver Operating Characteristic (ROC) curve. The area under this curve (AUC) indicates how well the model has separated the positive and negative classes.

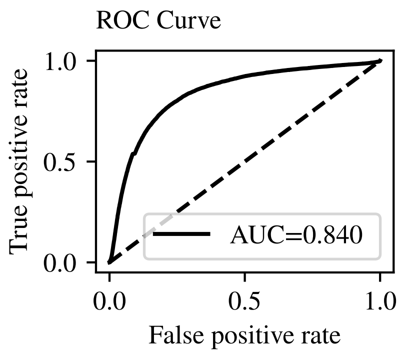
The graphs below are for an artificial dataset to illustrate the metrics. The histogram shows the number of elements of the positive and negative classes in ten ranges of p. The dataset is imbalanced like crash data, with far more crash persons not needing an ambulance than needing one.

The ROC curve is a parameterization over p of the true positive rate versus the false positive rate, with $p = 1$ in the lower left and $p = 0$ in the upper right. If the model perfectly separated the positive and negative classes, the parameterized curve would form a Gamma (Γ) from $(0, 0)$ to $(0, 1)$ to $(1, 1)$ given AUC $= 1$. Random noise would follow the dashed diagonal and give AUC $= 0.5$, as illustrated further below.

Ideal Model In this model, most of the negative class samples are on the left (TN, $p < 0.5$) and most of the positive class samples are on the right (TP, $p > 0.5$), which is what we want. Under this model, we would immediately dispatch ambulances to 77.7% of the crash persons who needed one (recall), and 38.7% of the ambulances we immediately dispatched would be needed (precision). The harmonic mean of those two numbers is F1 = 0.517. The area under the curve, which increases with how well the model has separated the two classes, is 0.840.

```
Ideal
```





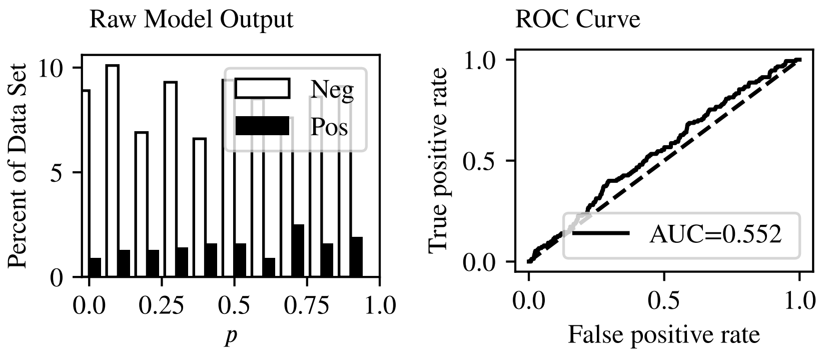
Prediction

Actual	N	P
N	66.5%	18.5%
P	3.3%	11.7%

0.781	Accuracy
0.387	Precision
0.777	Recall
0.517	F1
0.840	AUC

Awful Model A model with these results has not successfully separated the positive and negative classes, possibly because the data has no pattern.

```
Awful
```



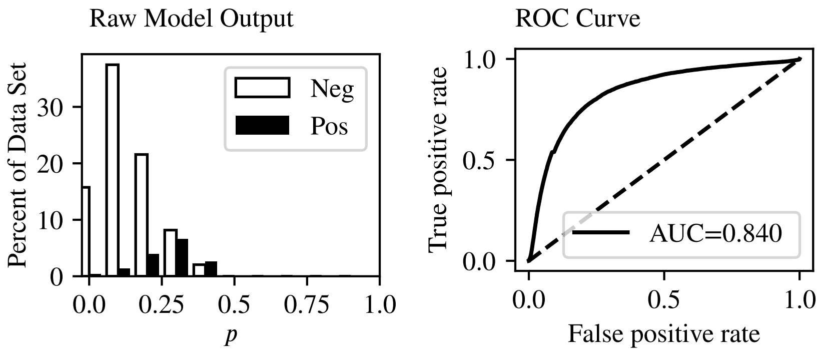
Prediction

Actual	N	P
N	41.8%	43.2%
P	6.5%	8.5%

0.503	Accuracy
0.164	Precision
0.567	Recall
0.248	F1
0.552	AUC

Ideal Model Tending Left Under this model, using decision threshold $\theta = 0.5$, we would immediately dispatch no ambulances. The model has separated the positive and negative classes just as well (AUC = 0.840), but the entire distribution is pushed to the left. We did encounter this situation in our work.

```
Ideal_Left
```



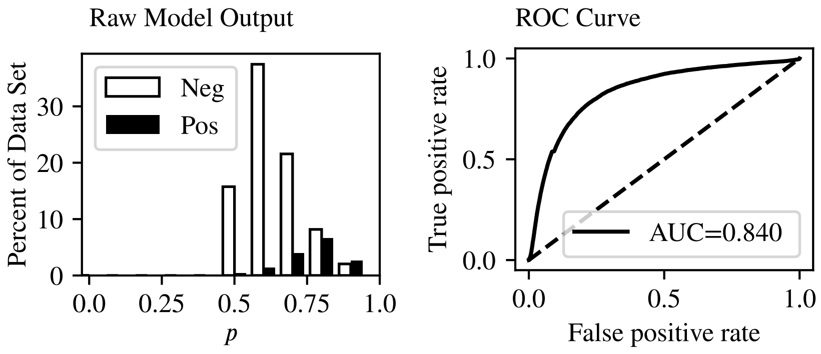
Prediction

Actual	N	P
N	85%	0%
P	15%	0%

0.85	Accuracy
nan	Precision
0	Recall
nan	F1
0.840	AUC

Ideal Model Tending Right Under this related model, if we were using decision threshold $\theta = 0.5$, we would immediately dispatch an ambulance to every reported crash, which would save lives but be expensive and not possible with existing resources in the short term.

```
Ideal_Right
```



Prediction

Actual	N	P
N	0%	85%
P	0%	15%

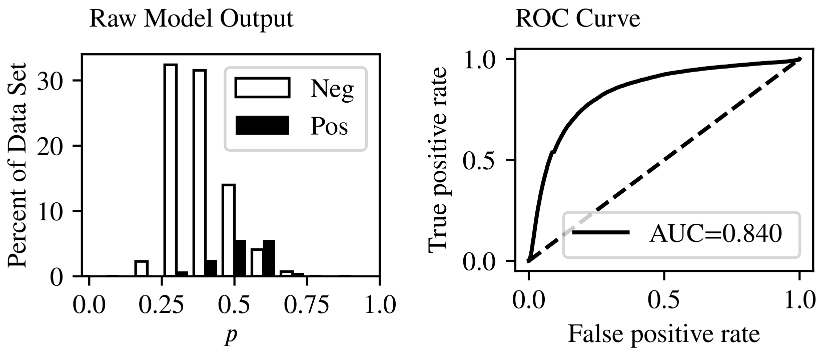
0.15	Accuracy
0.15	Precision
1	Recall
0.15	F1
0.840	AUC

If a model gives results like the "Ideal Model Tending Left," and we directly apply its recommendations to never immediately dispatch an ambulance, then we wasted our time building a recommendation system. The model output can still be useful, however, if we approach the data in one of three ways.

Shift the *p* Values to the Right

Here we have shifted all of the *p* values so that the average of the median *p* value for the negative class and the median *p* value for the positive class is at 0.5.

`Ideal_Left_Shifted`



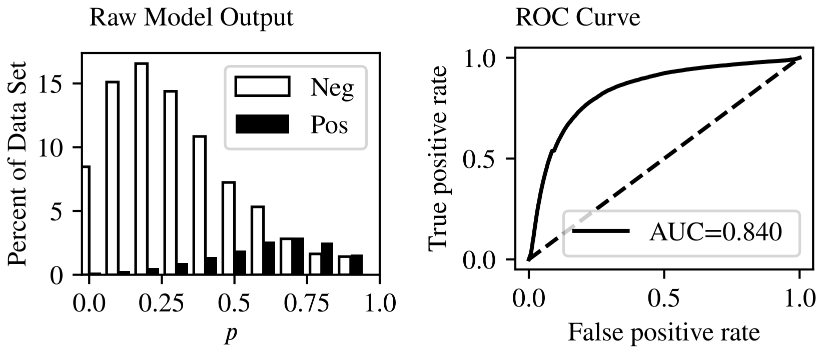
Prediction

Actual	N	P
N	66.3%	18.7%
P	3.30%	11.7%

0.780	Accuracy
0.385	Precision
0.780	Recall
0.516	F1
0.840	AUC

Linearly Transform the *p* Values, mapping the min to 0.0 and the max to 1.0. If there are far outliers, perhaps map the 1% quantile to 0 and the 99% quantile to 1.0. This transformation is especially useful for visualizing the data.

`Ideal_Left_Linear_Transform`



Prediction

Actual	N	P
N	66.6%	18.4%
P	3.37%	11.6%

0.783	Accuracy
0.388	Precision
0.775	Recall
0.518	F1
0.840	AUC

Note that the ROC curve does not change under these transformations.

Choose a Different Decision Threshold

The table below shows the number of elements of the negative and positive classes in bands of values of *p*.

`Ideal_Left_20`

p	Neg	Pos	mPrec	TN	FP	FN	TP	Prec	Rec	FP/P	$\hat{p}$
0.00	0	0	nan	0	85,000	0	15,000	0.15	1.00	5.67	1.00
0.05	2,481	137	0.05	2,481	82,519	137	14,863	0.15	0.99	5.50	0.97
0.10	13,268	260	0.02	15,749	69,251	397	14,603	0.17	0.97	4.62	0.84
0.15	19,408	469	0.02	35,157	49,843	866	14,134	0.22	0.94	3.32	0.64
0.20	18,054	941	0.05	53,211	31,789	1,807	13,193	0.29	0.88	2.12	0.45
0.25	13,262	1,599	0.11	66,473	18,527	3,406	11,594	0.38	0.77	1.24	0.30
0.30	8,302	2,338	0.22	74,775	10,225	5,744	9,256	0.48	0.62	0.68	0.19
0.35	5,514	3,245	0.37	80,289	4,711	8,989	6,011	0.56	0.40	0.31	0.11
0.40	2,660	3,354	0.56	82,949	2,051	12,343	2,657	0.56	0.18	0.14	0.05
0.45	1,352	2,152	0.61	84,301	699	14,495	505	0.42	0.03	0.05	0.01
0.50	699	505	0.42	85,000	0	15,000	0	nan	0.00	0.00	0.00
0.55	0	0	nan	85,000	0	15,000	0	nan	0.00	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1.00	0	0	nan	85,000	0	15,000	0	nan	0.00	0.00	0.00

Choosing decision thresholds $\theta = 0.20$ and $\theta = 0.25$ give us these confusion matrices.

$\theta = 0.20$		Prediction	
		N	P
Actual	N	53,211	31,789
	P	1,807	13,193

$\theta = 0.25$		Prediction	
		N	P
Actual	N	66,473	18,527
	P	3,406	11,594

To choose $\theta = 0.25$ instead of $\theta = 0.20$ is to send 1,599 fewer ambulances to crash persons who need them (more false negatives), but also 13,262 fewer ambulances to crash persons who did not need them (fewer false positives). Where to make that tradeoff of lives for money is a political question, not a technical one, but given the political decision we can choose θ to calibrate the recommendation system.

For instance, in the short term the number of available ambulances and ambulance crews is fixed. The FP/P column gives the amount of increase in the number of ambulances going to crash persons because we are immediately dispatching some ambulances that are not needed. If the existing infrastructure could handle only a 5% increase in the number of ambulances sent to crash persons, then we would choose $\theta = 0.45$, immediately dispatching 1,204 ambulances, 505 of which were needed and would have been sent anyway, but 699 of which are not needed, representing an increased cost. If we could handle a 10% increase, we would zoom in to the data to get a value of θ between 0.40 and 0.45 where FP/P = 0.10

This section repeats a lot of stuff from the Metrics section; put in Methods

The binary classification process actually has more nuance that can lead to more interesting metrics. The model does not tell us directly whether it predicts that the sample belongs to the positive or negative class; instead, the model assigns to each sample a value $p \in [0, 1]$. In code using scikit-learn, the value p is often called `y_proba` and is returned by the `model.predict_proba()` function. This value p is a proxy for the probability that the crash person needs an ambulance. It is not exactly the probability, but increases with the probability. If we have enough data to smooth out the effects of randomness, and if there actually is an underlying pattern to the data that will predict the target variable, then we can can get rough probabilities from p and measure marginal effects of changing the decision threshold θ.

To map the probability that a crash person needs an ambulance as a function of p, we looked at ranges of p of width 0.01, and in each band divided the number of positive samples in the band by the total number of samples in the band. We would like to call this metric "marginal precision," but that term is used in statistics for something else. It is the marginal probability, the probability that the a crash report with $p = \theta$ needs an ambulance, where Precision is the total probability, the proportion of crashes with $p > \theta$ that need an ambulance.

To get enough samples in each band to smooth out the randomness, we needed more than the usual 70/30 train/test split, so we went to 5-fold cross validation. In this approach, we divide the dataset into five sets, each with the same P/N ratio. We then train the model five times with a 80/20 train/test split, which gives us a p value for each sample in the whole set.

There are two reasons why we chose a width of 0.01 for our p bands. First, with a much smaller width we would have far fewer samples in each band, showing more of the effects of randomness. Second, some of our model algorithms, like the Balanced Random Forest Classifier from Imbalanced-Learn, gave most (93%) of the p values rounded to two decimal places. Intervals like $p \in [0.501, 0.509)$ would have few samples, and thus not be useful. The Balanced Bagging model algorithm gave 99% of the values of p rounded to one decimal place, so for that model we would need to choose a band width of 0.10.

2.3. The Dataset

Ideally, we would use a dataset of crashes with 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.

We will use the Crash Report Sampling System (CRSS) 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, between 2 and 3 percent. Since most 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 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.

We merged the `accident.csv`, `vehicle.csv`, and `person.csv` files in the six years. We dropped many features that were irrelevant, most because they were unique to each vehicle like a VIN (Vehicle Identification Number), with no predictive power, just random noise. We also dropped the 33,776 persons in crashes involving a pedestrian, because the deceleration profile of hitting a pedestrian or bicycle would not be different enough from hard braking to trigger an automated crash notification.

After removing repeated and irrelevant features and pedestrian crashes, we have 118 features describing 713,566 crash persons. Later we removed more features that have more than 20% of the values missing or only have data for some years, and features with imputed missing values, to get 78 features.

For full details, see the `Ambulance_Dispatch_01_Get_Data.ipynb` file.

2.4. Binning Categories

All of the CRSS data is discrete, but some features are ordered, like `HOUR` and `AGE`, and others are unordered, like `MAKE_MOD`. Reducing the dimensionality of the machine learning modeling by binning the categories into less than ten per feature is ideal.

Some features like `HOUR` we binned by hand. We looked at the proportion of crash persons hospitalized at each hour and found clear places to break it into seven contiguous but not equal blocks. When we looked at `AGE`, we considered breaking it into decades, but found that ages 15-18 have a far lower hospitalization rate than those a little younger or older, and there was a shift at about age 53 and again around age 74, so we binned it accordingly.

Some features like `MAKE_MODm` which has 1,210 unique values, we binned by imposing an order, ordering by the proportion of crash persons hospitalized, then cutting the ordered list into five new categories plus "Unknown."

For full details, see `Ambulance_Dispatch_02_Correlation.ipynb` and
`Ambulance_Dispatch_03_Bin_Data.ipynb`.

2.5. Imputing Missing Data

For reasons of historical consistency going back to 1982 with the predecessors of CRSS, CRSS imputes missing values for some features but not others, using IVEware, Imputation and Variation Estimation Software from the Institute for Social Research at the University of Michigan. Fortunately, when CRSS gives a feature with imputed values, it also retains the original feature with values signifying "Unknown." CRSS has a very helpful report on its imputation methods. We have not seen in the literature where someone has used IVEware to impute the other features and compared it to other methods.

At this point we have 78 unimputed features, and only 250,389 out of 713,566 samples (35%) did not have missing values in those 78 features. We compared three methods for imputing missing values.

- IVEware

- Imputation to Mode
- Round-Robin Random Forest using Imputation to Mode as the starting point

We found that the Random Forest method was best for our purposes.
For full details and analysis, see `Ambulance_Dispatch_04_Impute_Missing_Data.ipynb`.

2.6. Order of Operations

We also considered whether the order of operations made a difference: Should we bin first, then impute, or impute first, then bin?. We tried both methods on the `Ambulance` dataset with the IVEware imputation approach. After several runs we found that the difference between methods was about the same as the difference between runs of the same method with different random seeds. Since IVEware can only handle up to about forty categories in each categorical field, we had had to bin some fields first either way, so we chose to bin first, then impute.

For full details and analysis, see `Ambulance_Dispatch_05_IVEware_Order_of_Operations.ipynb`.

2.7. Handling Imbalanced Data

In our dataset only about fifteen percent of the people needed an ambulance. If a recommendation system never sent an ambulance, the model would have 85% accuracy, but be useless. Most algorithms for training models are designed for balanced data, with half of the samples in each of the negative and positive classes. With an imbalanced data set we can address the imbalance in five ways: Resampling the dataset, modifying the loss function, choosing metrics other than accuracy, using learning methods that account for the imbalance, and manually moving the decision threshold.

2.7.1. Resampling the Dataset

We can balance the dataset by undersampling the majority class (negative, “No ambulance”) or oversampling the minority class (positive, “Send Ambulance”). To balance by undersampling would mean throwing out eighty percent of the majority class, losing valuable information. A very popular method for oversampling is SMOTE (Synthetic Minority Oversampling TEchnique), which creates new minority samples between existing minority samples, but the “between” requires continuous data, and all of our data is discrete or categorical (What is between a Buick and a Volvo?).

Tomek Links is one of the few resampling methods that works for categorical data. It is a selective undersampling method that removes majority samples that seem out of place. We did not see a significant improvement in the model metrics from the undersampling; the difference between no undersampling, one run of Tomek, and two runs turned out to be inconsequential, by which we mean that one approach was not consistently better (measured by the area under the ROC curve) when we ran the models with different random seeds.

We considered a related method, Condensed Nearest Neighbor, but it is impractical for data sets of this size.

2.7.2. Modifying the Loss Function

A popular and well established way to modify the loss function for imbalanced data is with class weights, which can have the same effect as naïve oversampling.

Three of our seven models take class weights, and for those we tried three different class weights.

α	Meaning
1/2	No class weight
2/3	$\Delta FP/\Delta TP < 2.0$ goal
0.85	Balanced classes

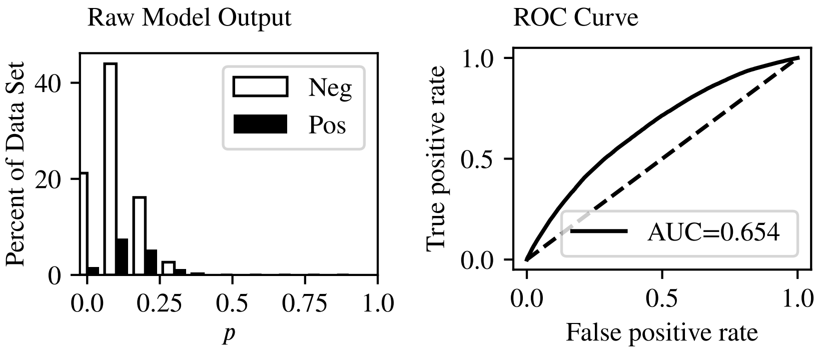
A related method is with focal loss, which has a modulating hyperparameter γ that increases the penalty for low-confidence samples. (Lin, Goyal, Girshick, He and Dollár, 2017) We tried five values of γ.

γ	Notes
0.0	Same as binary crossentropy
0.5	Very light modulation
1.0	Light modulation
2.0	Recommended by Lin
5.0	Heavy modulation

We did not see significant improvement using focal loss, measured by the area under the ROC curve. (**Put in Label Reference**).

2.7.3. Metrics for Imbalance

In the **??** subsection above we defined the metrics recall, precision, and f1. The most common metric in machine learning, the one that most algorithms are designed to maximize, is accuracy, the proportion of samples correctly classified. In that section's example of transformed model output, we had 150,107 out of 177,392 test samples correctly classified, giving 84.6% accuracy. Is that good? The model below, the raw results of the Logistic Regression model of the easy features set, recommends sending no ambulances, and it is correct in 150,771 of 177,392 test samples, giving 84.99% accuracy. Is that better?



Prediction

Actual	N	P
N	150,771	0
P	26621	0

0.8499	Accuracy
und	Precision
0.0	Recall
und	F1
0.659	AUC

In this study, we have arbitrarily decided that we are willing to trade off up to two false positives to get one more true positive. Once we moved our decision thresholds to the ethical tradeoff point, the accuracy only varied from 0.836 to 0.854. The difference in accuracy tells us how many more (or fewer) false positives than true positives we have, with them being equal at 0.8499, and we get the same information from precision being less than, more than, or equal to 0.5. Therefore, we are not going to consider accuracy in evaluating our models.

2.7.4. ML Algorithms for Imbalanced Data

[Expand this subsubsection]

- Random Undersampling Composite Models
- Bagging
- Boosting

2.8. Models

We used seven binary classification algorithms. Three of them take class weights.

Model	Source	Class Weights
KerasClassifier with the Binary Focal Crossentropy loss function	Keras	Yes
Balanced Random Forest Classifier	Imbalanced-Learn	Yes
Balanced Bagging Classifier	Imbalanced-Learn	No
RUSBoost Classifier	Imbalanced-Learn	No
Easy Ensemble Classifier with AdaBoost Estimator	Imbalanced-Learn	No
Logistic Regression Classifier	Scikit-Learn	Yes
AdaBoost Classifier	Scikit-Learn	No

For the focal loss function, we tried seven different combinations of the hyperparameters α for class weights and γ for penalty on badly misclassified samples. For the random forest and bagging models we tried three values of α. Altogether we had seventeen model/hyperparameter combinations. We learned each of the seven models on datasets with the easy, medium, and hard features, and on the hard features we tested with Tomek undersampling 0, 1, and 2 times, for a total of five datasets, giving eighty-five model/hyperparameter/dataset combinations. We learned each of those sixty-five with two different random seeds, for a total of one hundred seventy results.

Seventeen Models

Model	α	γ
Focal	1/2	0.0
Focal	2/3	0.0
Focal	2/3	0.5
Focal	2/3	1.0
Focal	2/3	2.0
Focal	2/3	5.0
Focal	0.85	0.0
Random Forest	1/2	
Random Forest	2/3	
Random Forest	0.85	
Bagging		
RUSBoost		
Easy Ens		
Log Reg	1/2	
Log Reg	2/3	
Log Reg	0.85	
AdaBoost		

$\times$

Seven Datasets

Features	Tomek
Hard	None
Hard	Once
Hard	Twice
Medium	None
Medium	Once
Medium	Twice
Easy	None

$\times$

Run twice with different random seeds
Random seed 1
Random seed 2

$=$ 238 Sets of Results

3. Results

4. Conclusions

5. Discussion

6. 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`

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

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.