

## Highlights

### **Modeling the Need for an Ambulance based on Automated Crash Reports from iPhones**

First Author, Second Author, Third Author, Fourth Author

- 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 the Crash Report Sampling System.
- 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.
- Perennial Machine Learning Challenge: Imbalanced Datasets.

# Modeling the Need for an Ambulance based on Automated Crash Reports from iPhones

First Author<sup>a,b</sup>, Second Author<sup>a</sup>, Third Author<sup>a,c</sup> and Fourth Author<sup>c</sup>

<sup>a</sup>School, University,

<sup>b</sup>Other School,

<sup>c</sup>Other Department, University,

---

## ARTICLE INFO

### Keywords:

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

## ABSTRACT

New Google Pixel phones can automatically notify police if the phone detects the deceleration profile of a crash. From the data available from such an automatic notification, can we build a machine-learning model that will recommend whether police should immediately, perhaps automatically, dispatch an ambulance? If the injuries are serious, time to medical care is critical, but few crashes result in serious injuries, and ambulances are in limited supply and expensive.

The costs of the false positives and false negatives are very different. The cost of sending an ambulance when one is not needed is measured in dollars, but the cost of not sending an ambulance when one is needed is measured in lives. Each society chooses a marginal ethical tradeoff rate  $\omega = \Delta FP / \Delta TP$  when it sets government budgets. For our work we arbitrarily chose  $\omega = 2.0$  and incorporated it into the model in the class weight and decision threshold.

We will show that the quality of the model depends mostly on what information is available to inform the decision of whether to immediately dispatch an ambulance. We used the data of the Crash Report Sampling System (CRSS), which is freely available online. 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. Literature Review

## 3. Dataset

## 4. Methods

### 4.1. Metrics

**Precision** tells us, of the ambulances we sent, how many were needed. **Recall** tells us, of the ambulances that were needed, how many we sent. Recall only looks at elements of the minority class (positive class, “need ambulance”), so is independent of the class imbalance. Precision is affected by class imbalance, but is still relevant to our decisions in its imbalanced form. Because the number of elements of the positive class in the test set is constant across all of our models, recall is proportional to TP.

The **F1 score** is the harmonic mean of precision and recall. Why the harmonic mean instead of the arithmetic or geometric? For two positive numbers  $a$  and  $b$  with  $0 < a < b$ ,


$$a < Harm(a, b) < Geo(a, b) < Arith(a, b) < b$$

so the F1 score emphasizes what the model does poorly. We will use F1 as our primary indicator, while looking at precision and recall.

The area under the curve (**AUC**) of the receiver operating characteristic (ROC) is a measure of how well a model separates the samples of the positive and negative classes. We will use it to show that the additional features in the “hard/expensive” and “medium” datasets are important for discriminating between the two classes.

The  $\Delta FP / \Delta TP$  curve is related to the ROC;  $\Delta FP / \Delta TP$  is the reciprocal of the product of the slope of the ROC curve and a factor that corrects for class imbalance.

---

 FirstAuthor@gmail.com (F. Author)  
ORCID(s):

$$\frac{\Delta FP}{\Delta TP} = \frac{N}{P} \cdot \frac{\frac{\Delta FP}{N}}{\frac{\Delta TP}{P}} = \frac{N}{P} \cdot \frac{\Delta FPR}{\Delta TPR} = \frac{1}{\frac{P}{N} \cdot \frac{\Delta TPR}{\Delta FPR}} = \frac{1}{\frac{P}{N} \cdot mROC}$$

We will use this curve to find the value of the discrimination threshold where  $\Delta FP/\Delta TP = 2.0$

## 4.2. Incorporating the $\Delta FP/\Delta TP < \omega$ Ethical Threshold

We incorporated this ethical threshold in two ways, as a class weight and as the decision threshold.

### 4.2.1. Class Weight

To understand why class weights can encode this threshold, we will use the  $\alpha$ -weighted binary crossentropy model as an example.

To move the model towards  $\Delta FP/\Delta TP < \omega \rightarrow \Delta FP - \omega \Delta TP < 0$  is equivalent to minimizing  $FP - \omega TP$ . The  $\alpha$ -weighted binary crossentropy loss function is

$$Loss = - \left( \alpha \sum_{y=1} \log(h_{\theta}(x_i)) + (1 - \alpha) \sum_{y=0} \log(1 - h_{\theta}(x_i)) \right)$$

In this function,  $y_i \in \{0, 1\}$  is the ground truth for sample  $i$ , 0 if in the majority class ("no ambulance") and 1 if in the minority class ("yes ambulance"). The term  $h_{\theta}(x_i) \in [0, 1]$  is the probability that  $x_i$  is in the minority class, as calculated by this  $\theta$  iteration of the model.

The TN, FP, FN, and TP are discrete, and the  $h_{\theta}(x_i)$  is continuous. To see how they relate, let us discretize the loss function, with

$$\log(h_{\theta}(x_i)) \rightarrow \begin{cases} 0 & \text{if } h_{\theta}(x_i) \leq 0.5 \\ 1 & \text{if } h_{\theta}(x_i) > 0.5 \end{cases} \quad \text{and} \quad \log(1 - h_{\theta}(x_i)) \rightarrow \begin{cases} 0 & \text{if } 1 - h_{\theta}(x_i) \leq 0.5 \\ 1 & \text{if } 1 - h_{\theta}(x_i) > 0.5 \end{cases}$$

which makes  $\sum_{y=1} \log(h_{\theta}(x_i))$  into  $TP$  and  $\sum_{y=0} \log(1 - h_{\theta}(x_i))$  into  $TN$ , making the discrete version of the loss function

$$Loss = -(\alpha TP + (1 - \alpha)TN)$$

In the following manipulations, note that adding a constant, or multiplying by a positive constant, does not change the effect of the loss function, which the model algorithm uses to compare one iteration to the next.

$$Loss = -(\alpha TP + (1 - \alpha)TN)$$

$$Loss = -\left(\frac{\omega}{\omega + 1}TP + \frac{1}{\omega + 1}TN\right) \quad \text{Let } \alpha = \frac{\omega}{\omega + 1}, \text{ making } 1 - \alpha = \frac{1}{\omega + 1}$$

$$Loss = -(\omega \cdot TP + TN) \quad \text{Multiply by } \omega + 1$$

$$Loss = -(\omega \cdot TP + TN) + TN + FP \quad \text{Add constant } TN + FP, \text{ the number of majority samples}$$

$$Loss = FP - \omega \cdot TP$$

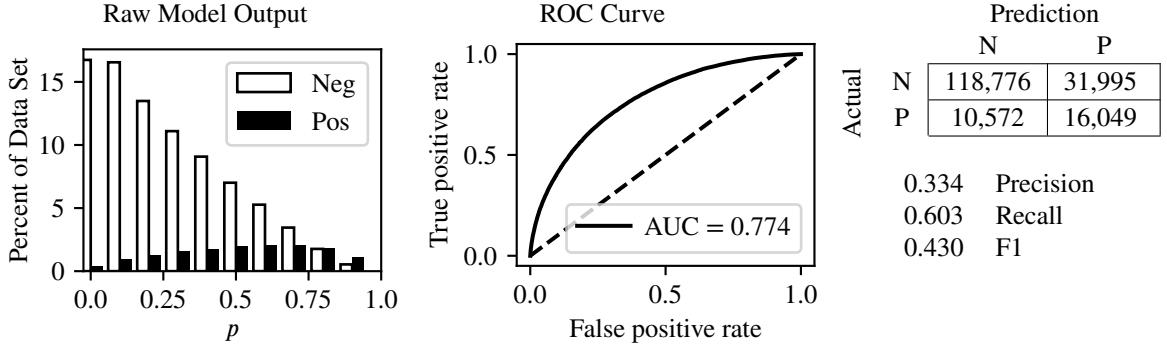
Thus, we can incorporate the ethical threshold  $p$  into our loss function as the class weight. For this study we have arbitrarily chosen  $\omega = 2$ , so we will use class weight  $\alpha = \omega/(\omega + 1) = 2/3$ .

### 4.2.2. Decision Threshold

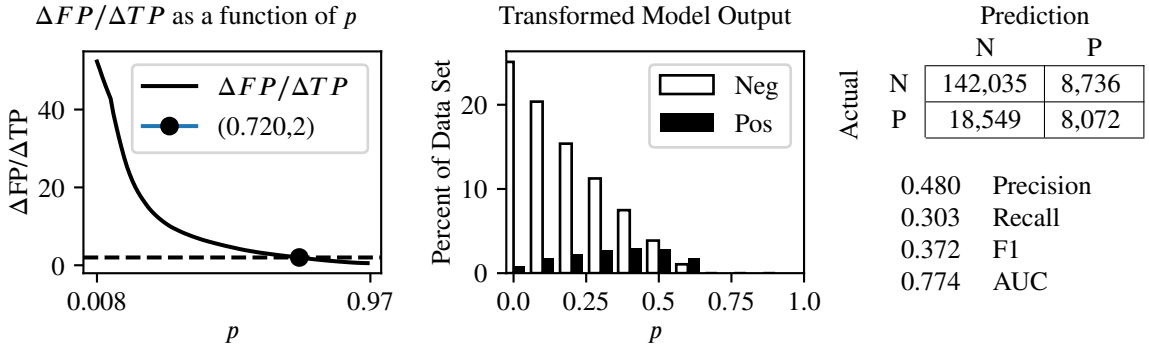
Once a supervised learning algorithm learns a model using the training set, it evaluates the model on the test set, returning for each sample in the test set a probability  $p \in [0, 1]$  that the sample belongs to the positive class ("need ambulance"). By default we use a decision threshold  $p = 0.5$  to discriminate between predicted negative (PN) and predicted positive (PP), but for good cause we can choose a different threshold. We will choose to make the decision threshold the value of  $p$  that makes  $\Delta FP/\Delta TP = 2$ . Because many tools are built around having the decision threshold

at  $p = 0.5$ , rather than change the decision threshold we will linearly transform the probabilities  $p \in [0, 1]$  to shift the desired threshold of  $p$  where  $\Delta FP/\Delta TP = 2$  to  $p = 0.5$ .

Consider these results from one of our models. The histogram shows, for the negative (“no ambulance”) and positive (“ambulance”) classes, the percentage of the dataset in each range of  $p$ . In an ideal model, the negative class would be clustered on the left and the positive on the right. The ROC curve and the area under the curve (AUC) indicates how cleanly the model separates the two classes, with  $AUC = 1$  being perfect and  $AUC = 0.5$  being basically random classification.



Mapping  $\Delta FP/\Delta TP$  as a function of  $p$  for this model, we see that it equals  $\omega = 2$  when  $p = 0.720$ . Using a linear transformation, we can map 0.720 to 0.5, keeping 0 at 0, to get transformed model output with the decision threshold where  $\Delta FP/\Delta TP = 2$ . The ROC curve and its AUC are invariant under such a transformation.



It is reasonably to ask, “How is the transformed model ethically better? We are only sending 8,072 needed ambulances instead of 16,049. In the original model,  $FP/TP = 31,995/16,049 = 1.994 < 2.0 = \omega$ . How is it better to send half as many needed ambulances?” Because our ethical tradeoff was not for total number of FP and TP, but for marginal FP and TP. Going from the original to the transformed model, we have  $\Delta FP/\Delta TP = (31,995 - 8,736)/(16,049 - 8,072) = 23,259/7,977 = 2.91$ , which is higher than our choice of ethical tradeoff  $\omega = 2.0$ . For this model, it is at  $p = 0.720$  that we reach our tradeoff point.

We will use the model outputs transformed to have decision threshold at  $\Delta FP/\Delta TP = 2.0$  to compare different models.

### 4.3. Preparing the Data

The CRSS data is available [online at this link](#). The three main files for each year are Accident, Vehicle, and Person, and one uses the CASENUM and VEH\_NO fields to merge them into one dataset.

#### 4.3.1. Order of Operations

To prepare the data we needed to do two things, to bin (discretize) some features and to impute missing data. We did not know which to do first, so we tested both ways using IVEware (Raghunathan, Solenberger, Berglund and van Hoewyk) for the imputation. The imputation is a stochastic process, and the difference between binning first and imputing first was as small as the difference between running twice 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 decided on binning first.

#### 4.3.2. Binning

To bin a field's many categories into fewer categories, sometimes the meaning of the categories was a sufficient guide. In the HOSPITAL field, which we used as our target variable, we were only interested in two values, whether or not the person went to the hospital. The CRSS field has six values indicating how the person went to the hospital (ground ambulance, air ambulance, ...), and we merged those into one. For fields where the binning was not so obvious, we looked at how each value in the field correlates to hospitalization. We wanted to put AGE into bands, and looked to divide where the hospitalization rate changed. Interestingly, ages 16, 17, and 18 have lower hospitalization rates than ages below or above, so we put them into their own band. Around age 52 the hospitalization rate started to go up, so we split there. We binned other fields in a similar way.

The merging, dropping, and binning are all in the CRSS\_04\_Discretize code.

#### 4.3.3. Imputing Missing Values

About 47% of the samples had unknown values in the thirty-eight fields we use for our analysis. The CRSS authors imputed unknown values in ten of those fields, another seventeen had no unknown values, but eleven fields we want to use had missing values that were not imputed by CRSS. The CRSS authors have a very helpful report on their imputation methods. (Herbert, 2019) The reasons why some fields get imputed include historical consistency going back to 1982.

(See CRSS\_04\_5\_Count\_Missing\_Values)

When the CRSS authors imputed unknown values for a field, they published two fields, one with the imputed values and one with the values signifying "Unknown." We discarded the imputed fields and compared three methods for imputing missing values. Impute to Mode assigns to all missing values in a feature the most common value in that feature. IVEware: Imputation and Variance Estimation Software employs multivariate sequential regression, and is the method the CRSS authors used. Round Robin Random Forest, like in MissForest, was consistently the most accurate. We tested the methods by dropping all samples with missing values, randomly deleting (but keeping a copy of) fifteen percent of the known values, imputing, and comparing to the ground truth.

(See CRSS\_05\_Impute\_Random\_Forest for details.)

We did not address the question of incorrect data.

#### 4.4. Selecting Features

We selected three groups of features to see whether more information would improve the model.

The first group of features held information that the police would already know before receiving a crash notification, like time of day, day of week, and urban/rural. A crash on a Saturday night in a rural area is far more likely to need an ambulance than one in a city at rush hour, so if no information specific to the crash is available, how well can we predict whether an ambulance is needed? We thought of this set of features as "easy" or "baseline."

The second group of features also included specific location and the age and sex of the primary user of the phone. Is the vehicle in an intersection or in a parking lot? Did the car end up off the roadway? What is the speed limit on that road? Getting that information from the latitude and longitude in the automated report would require instantaneous correlation with detailed maps. Whether such information significantly improves the model will inform whether policymakers should invest the time and effort to have that information available. We thought of this information as "medium" in cost.

The "hard" or "expensive" features would require regularly updated maps (work zones, lighting conditions), correlating records to guess which car the cell phone user is driving, and correlating multiple cell phone reports to count how many people are involved.

We dropped all crashes with a pedestrian, because unlike a tree or other vehicle, hitting a pedestrian may not cause the sudden deceleration that a cell phone could distinguish from sudden braking, so the cell phone likely would not register it as a crash.

(See CRSS\_06\_Build\_Model for details.)

## 4.5. Handling Imbalanced Data

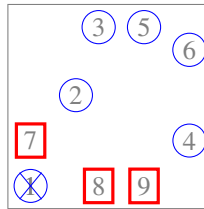
In our dataset only about fifteen percent of the people needed an ambulance. If a recommendation system never sent an ambulance, the model would have 85% accuracy, but be useless. Most algorithms for training models are designed for balanced data, with half of the samples in each of the negative and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

### 4.5.1. 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. A Tomek Link is a majority/minority pair that are each others’ nearest neighbors, which was the case with about four percent of the majority samples. We used the Tomek algorithm to remove the majority sample of each Tomek link, undersampling the majority class, and then running it again to remove more that had not been Tomek links in the first undersampling run.

Consider this two-dimensional training dataset. The six blue circles represent samples (elements) of the majority negative class (“no ambulance”), and the three red squares represent the minority positive class (“ambulance”). Samples #7 and #1 are each others’ nearest neighbors of different classes, so they are Tomek Links and the algorithm deletes #1. In a second Tomek run, once #1 is gone, #7 and #2 are Tomek Links, so the method deletes #2.



Our original dataset has 619,027 samples. We first removed the 27,723 crashes involving a pedestrian, leaving 591,304 samples. Each sample had 82 features; we cut the number of features to 38 for our “Hard” features, then to 21 for “Medium,” and to 10 for “Easy.” We then split each of those three datasets 70/30 into a training set of 413,913 samples and a test set of 177,393 samples, preserving the proportions of negative and positive samples in both sets. We did the train/test split twice with different random seeds (“Round 1” and “Round 2”) to gauge how much of the small differences in results were due to stochasticity instead of differences in the model algorithms or hyperparameters. Tomek undersampling only applies to the training set, not to the test set.

We then ran Imbalanced-Learn’s TomekLinks algorithm, then ran it again on the results to give our “Tomek Once” and “Tomek Twice” undersampled datasets.

Hard Features, Round 1				Hard Features, Round 2			
	Samples	Change			Samples	Change	
Original	413,913			Original	413,913		
Tomek Once	399,515	14,398	3.48%	Tomek Once	399,714	14,199	3.43%
Tomek Twice	396,511	3,004	0.75%	Tomek Twice	396,718	2,996	0.75%
Total Change		17,402	4.23%	Total Change		17,195	4.18%

Medium Features, Round 1				Medium Features, Round 2			
	Samples	Change			Samples	Change	
Original	413,913			Original	413,913		
Tomek Once	406,691	7,222	1.74%	Tomek Once	406,781	7,132	1.72%
Tomek Twice	405,288	1,403	0.34%	Tomek Twice	405,368	1,413	0.35%
Total Change		8,625	2.08%	Total Change		8,545	2.07%

Easy Features, Round 1				Easy Features, Round 2			
	Samples	Change			Samples	Change	
Original	413,913			Original	413,913		
Tomek Once	413,909	4	0.00097%	Tomek Once	413,908	5	0.00121%
Tomek Twice	413,908	1	0.00024%	Tomek Twice	413,907	1	0.00024%
Total Change		5	0.00121%	Total Change		6	0.00145%

We ran the models on the two rounds of Tomek undersampled training for the Hard-feature and Medium-feature sets, not for the Easy because the undersampling was so small.

We were disappointed to not see a significant improvement in the model metrics from the undersampling; the difference between no undersampling, one runs of Tomek, and two runs turned out to be inconsequential, by which we mean that one approach was not consistently better when we ran the models with different random seeds.

#### 4.5.2. 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. The Tomek undersampling changes the last weight slightly from 0.8499 to as low as 0.8433.

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

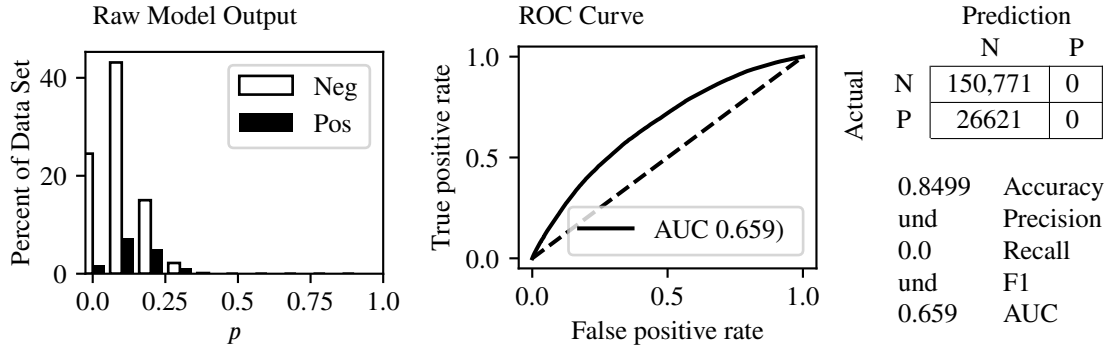
A related method is with focal loss, which has a modulating hyperparameter  $\gamma$  that increases the penalty for low-confidence samples. (Lin, Goyal, Girshick, He and Dollár, 2017) We tried five values of  $\gamma$ .

$\gamma$	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. (**Put in Label Reference**).

#### 4.5.3. Metrics for Imbalance

In the [Metrics](#) 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?



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.

#### 4.5.4. ML Algorithms for Imbalanced Data

[Expand this subsubsection]

- Random Undersampling Composite Models
- Bagging
- Boosting

#### 4.6. 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  $\alpha$  for class weights and  $\gamma$  for penalty on badly misclassified samples. For the random forest and bagging models we tried three values of  $\alpha$ . 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			Seven Datasets				
Model	$\alpha$	$\gamma$	Features	Tomek			
Focal	1/2	0.0	Hard	None	Run twice with different random seeds = 238 Sets of Results		
Focal	2/3	0.0		Once			
Focal	2/3	0.5		Twice			
Focal	2/3	1.0		None			
Focal	2/3	2.0		Once			
Focal	2/3	5.0		Twice			
Focal	0.85	0.0		None			
Random Forest	1/2	×	Hard	Twice	×	Random seed 1 Random seed 2	
Random Forest	2/3		Medium	None			
Random Forest	0.85		Medium	Once			
Bagging			Medium	Twice			
RUSBoost			Easy	None			
Easy Ens							
Log Reg	1/2						
Log Reg	2/3						
Log Reg	0.85						
AdaBoost							

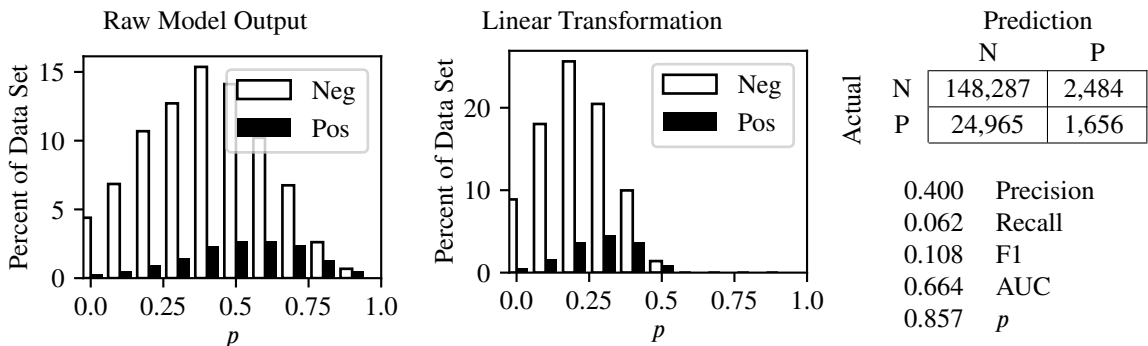
## 5. Results

### 5.1. More Features Improve the Model

Our model metrics show that having more information about each crash and crash person greatly increases the precision and recall of the model. The graphs and metrics below show the model with the highest F1 score for each of the easy, medium, and hard sets of features. In terms of F1, all of the easy-feature models were worse than all of the medium-feature models, and except for the Easy Ensemble Classifier models, all of the medium-feature models were worse than the hard-feature models. Full details about features for each set is in `CRSS_06_Build_Model_01_15_22.ipynb`.

The Easy set of features assumes that the automated crash notification comes to the police only with a location and that the police do not correlate that location information with detailed maps for information like whether that location is in a parking lot or a high-speed intersection. These features (day of week, time of day, weather,...) are information the police already had before the notification. This model balanced bagging model does correctly tell the police to dispatch 1,656 ambulances immediately with 40.0% precision, which is better than zero.

#### Easy Features: Balanced Random Forest model, No Tomek undersampling, No class weights

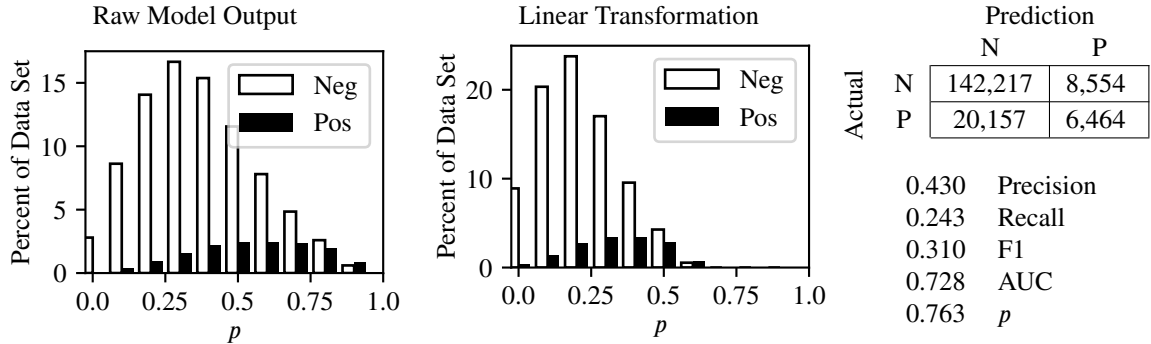


The Medium features assume two things.

1. The police can correlate the location with detailed maps to get, for instance, the kind of road and whether the crash was in an intersection
2. The automated cell phone crash report comes to the police from the cell service provider with some information about the likely user of the phone, including age and sex.

Our best model on the Medium features sends 6,464 needed ambulances with 43.0% precision, much better than the Easy features. Using this improved recall and precision is a good argument for investing in the data infrastructure necessary to make this information instantaneously available.

#### Medium Features: Balanced Radom Forest model, Tomek Once, No class weights

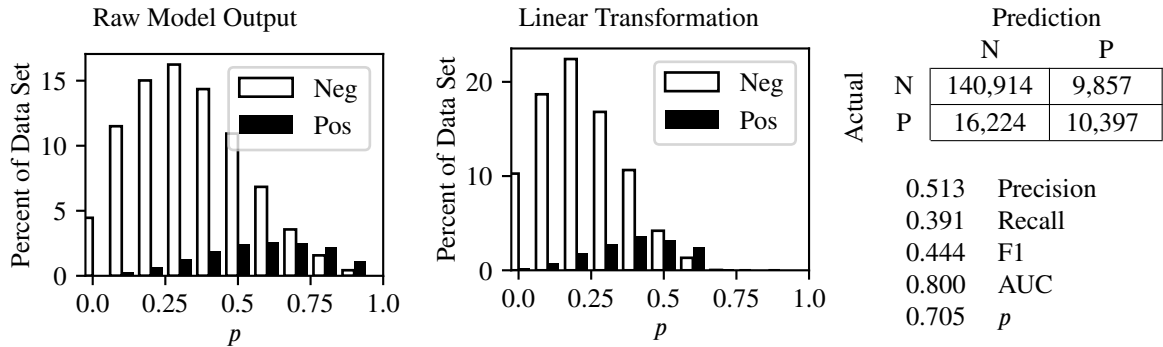


The Hard features assume that the police can do an additional three things.

1. Correlate information about the likely phone user with public and private records about automobile ownership and insurance to guess what kind of vehicle the person is driving
2. Correlate location with detailed and up-to-date maps of lighting conditions and work zones.
3. Correlate multiple notifications from the same location to determine (minimum) number of passengers, including whether a school bus is involved.

Our best model on the Hard features sends 10,397 needed ambulances with 51.3% precision. Again, this improvement in both precision and recall are good arguments for investing in the needed additional data infrastructure.

#### Hard Features: Balanced Random Forest model, Tomek twice, No class weights



## 5.2. Results of Undersampling and Hyperparameters

The table below shows the top thirty-two runs measured by the F1 score, and the top run for each of the five remaining model algorithms. All of the results are for the hard features.

The second column tells how many times we ran Tomek undersampling. The third column gives the values of the hyperparameters  $\alpha$  (class weight) that we used for the Balanced Random Forest, Keras Binary Focal Loss classifier, and Logistic Regression classifiers. The fourth column gives the values of  $\gamma$  (focal weight) we varied for the Keras binary focal loss classifier. We ran the models twice with different random seeds ("Round 1" and "Round 2" below), to see whether small differences were due to different models and hyperparameters or due to the inherent stochasticity of the train/test split and machine learning algorithms.

Algorithm	Tomek	$\alpha$	$\gamma$	Round	Precision	Recall	F1	AUC
BRFC	2	1/2		1	0.5133	0.3906	0.4436	0.8002
BRFC	1	1/2		1	0.5136	0.3880	0.4420	0.7998
BRFC	0	1/2		1	0.5145	0.3858	0.4410	0.7993
BRFC	2	2/3		1	0.5064	0.3904	0.4409	0.7997
BRFC	0	2/3		1	0.5107	0.3866	0.4401	0.7995
BRFC	0	0.85		1	0.5056	0.3892	0.4399	0.7986
BRFC	1	2/3		1	0.5093	0.3866	0.4396	0.7994
BRFC	2	1/2		2	0.5061	0.3857	0.4378	0.7982
BRFC	2	0.85		1	0.5054	0.3823	0.4353	0.7984
BRFC	1	1/2		2	0.5069	0.3810	0.4350	0.7977
BRFC	1	2/3		2	0.5041	0.3824	0.4349	0.7981
BRFC	1	0.85		2	0.5036	0.3801	0.4332	0.7967
BRFC	1	0.85		1	0.5185	0.3673	0.4300	0.7978
BRFC	2	0.85		2	0.5078	0.3702	0.4282	0.7970
BRFC	0	1/2		2	0.5173	0.3649	0.4280	0.7967
BRFC	0	2/3		2	0.5166	0.3652	0.4279	0.7977
BRFC	2	2/3		2	0.5123	0.3667	0.4274	0.7973
BRFC	0	0.85		2	0.5091	0.3652	0.4253	0.7961
KBFC	1	1/2	0.0	1	0.4825	0.3270	0.3898	0.7758
KBFC	0	1/2	0.0	1	0.4850	0.3196	0.3853	0.7753
KBFC	1	2/3	0.0	1	0.4848	0.3164	0.3829	0.7758
KBFC	0	2/3	0.5	2	0.4757	0.3202	0.3828	0.7755
KBFC	0	2/3	0.0	2	0.4790	0.3187	0.3827	0.7750
Bagging	2			1	0.5043	0.3079	0.3823	0.7664
KBFC	0	2/3	1.0	2	0.4805	0.3166	0.3817	0.7748
KBFC	1	0.85	0.0	1	0.4722	0.3199	0.3815	0.7742
KBFC	2	2/3	0.0	1	0.4830	0.3152	0.3814	0.7744
KBFC	2	2/3	0.5	1	0.4912	0.3117	0.3814	0.7751
KBFC	2	1/2	0.0	1	0.4886	0.3125	0.3812	0.7743
KBFC	2	2/3	1.0	1	0.4874	0.3127	0.3810	0.7750
KBFC	1	2/3	1.0	1	0.4855	0.3131	0.3807	0.7751
KBFC	1	2/3	0.5	2	0.4747	0.3171	0.3802	0.7740
$\vdots$								
LRC	1	1/2		1	0.4408	0.2895	0.3495	0.7548
AdaBoost	2			1	0.4408	0.2707	0.3354	0.7542
RUSBoost	0			1	0.4412	0.2660	0.3319	0.7547
EEC	0			1	0.4202	0.2488	0.3125	0.7329

The results indicate that Tomek undersampling and focal loss do not make a significant difference, especially compared with the much larger difference between the worst Balanced Random Forest Classifier model and the best Keras Binary Focal Crossentropy model. Though varying the hyperparameters did not give the most significant changes in the results, we found interesting behaviors that may be relevant to future research. What makes our look at focal loss different from most study is that we are only interested in the right tail past where  $\Delta FP/\Delta TP = 2.0$ .

### 5.3. Varying Binary Focal Crossentropy Hyperparameters

The Binary Focal Crossentropy loss function for Keras's neural network algorithm takes both a class weight hyperparameter  $\alpha$  and a modulating factor hyperparameter  $\gamma$  that gives more weight to samples badly misclassified. We tried combinations, including the values of  $\gamma$  tested in the paper Lin et al. (2017) The balanced class weight will vary slightly with undersampling of the majority class.

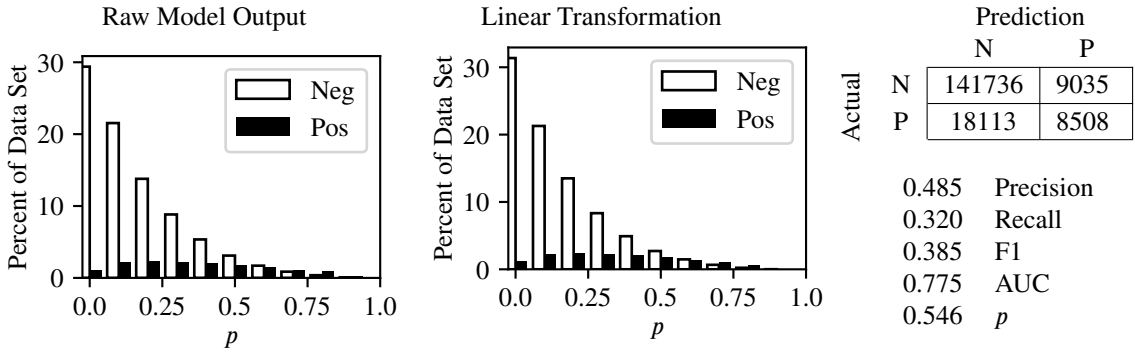
$\alpha$	Meaning	$\gamma$	Notes
1/2	No class weight	0.0	Same as binary crossentropy
2/3	$\Delta FP/\Delta TP < 2$	0.5	Very light modulation
0.85	Balanced class weight	1.0	Light modulation
		2.0	Recommended by Lin
		5.0	Heavy modulation

### 5.3.1. Varying Class Weights

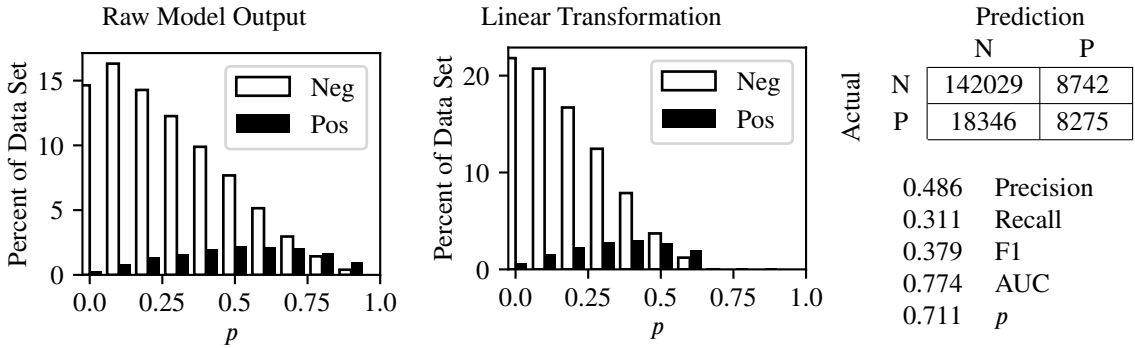
All three models used no Tomek undersampling and  $\gamma = 0.0$ .

With  $\alpha = 0.5$ , the model classifies the negative class well, but the positive class almost randomly. With  $\alpha = 0.85$ , balanced class weights, the model classifies the positive class well, but the positive class so poorly that the model does not separate the two classes well.

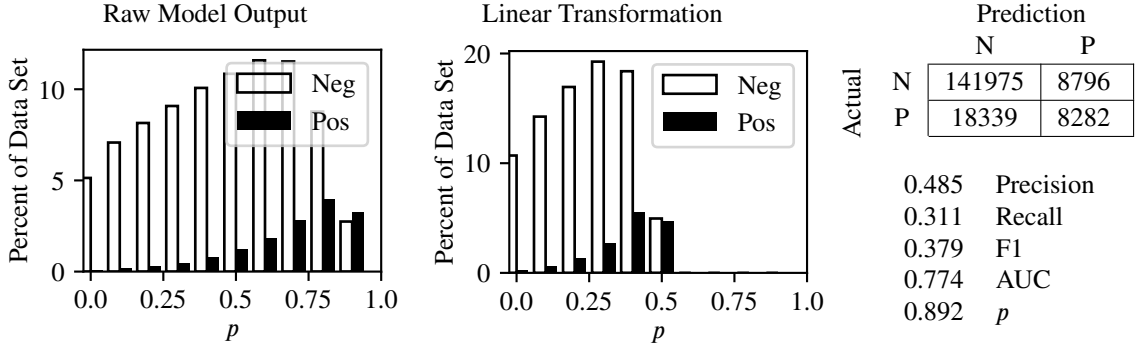
No class weight ( $\alpha = 0.5$ )



Class weight for  $\Delta FP/\Delta TP = 2.0$  ( $\alpha = 2/3$ )



Class weight for class balance ( $\alpha \approx 0.85$ )

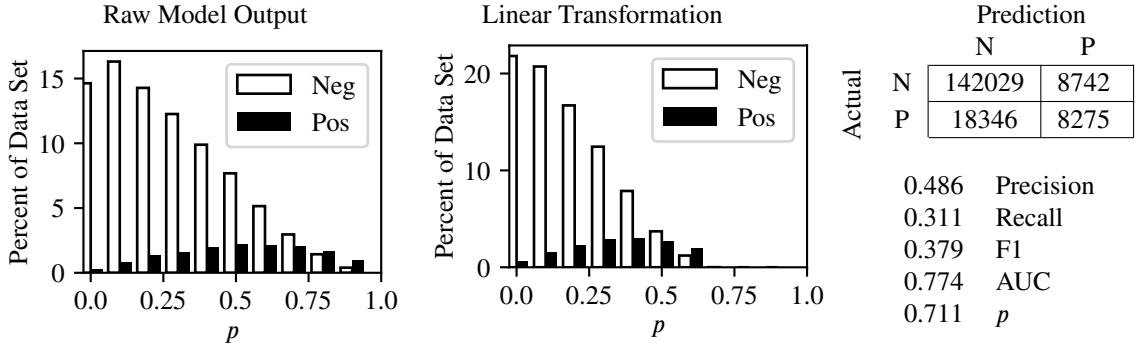


### 5.3.2. Varying $\gamma$ for Focal Loss

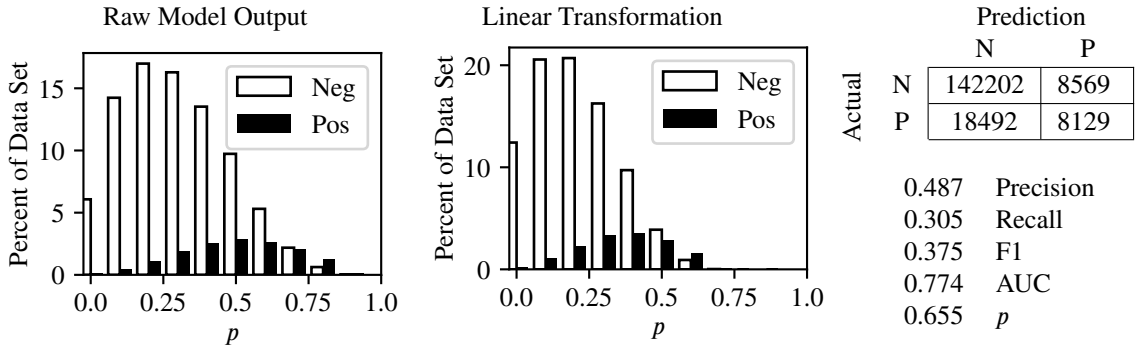
All five models used the class weight for  $\Delta FP/\Delta TP = 2.0$ ,  $\alpha = 2/3$ , with no Tomek undersampling.

The aspect that caught our attention was the way larger focal weights make the probabilities clusters towards the center, rather than the desired effect of giving more strongly classified samples.

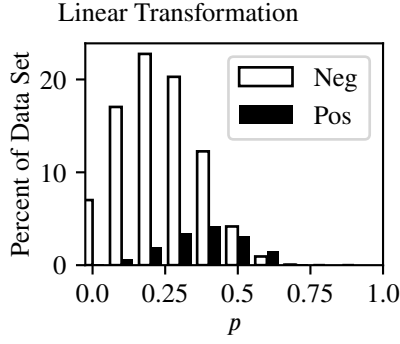
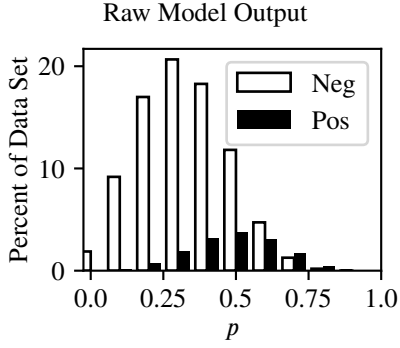
$\gamma = 0.0$ , same as non-focal binary crossentropy



$\gamma = 0.5$

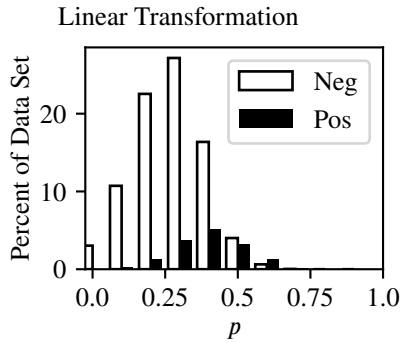
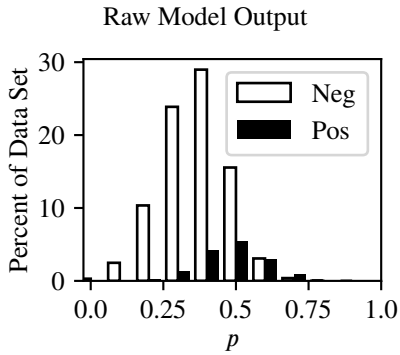


$\gamma = 1.0$



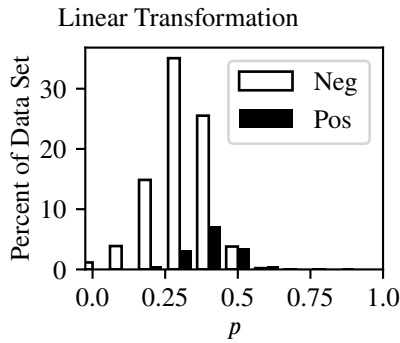
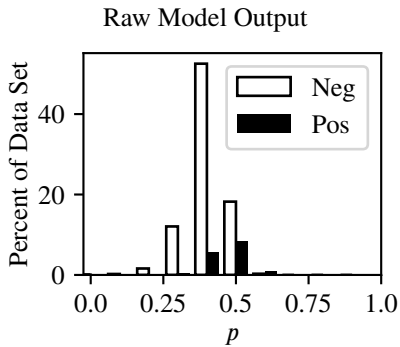
	Prediction	
	N	P
Actual N	141630	9141
Actual P	18242	8379
	0.478	Precision
	0.315	Recall
	0.380	F1
	0.775	AUC
	0.615	$p$

$\gamma = 2.0$



	Prediction	
	N	P
Actual N	142496	8275
Actual P	18602	8019
	0.492	Precision
	0.301	Recall
	0.374	F1
	0.774	AUC
	0.585	$p$

$\gamma = 5.0$



	Prediction	
	N	P
Actual N	143657	7114
Actual P	19351	7270
	0.505	Precision
	0.273	Recall
	0.355	F1
	0.773	AUC
	0.543	$p$

## 6. Conclusions

## 7. Discussion

## 8. Future Work

### Funding Statement

### Conflict of Interest

The authors have no relevant financial or non-financial interests to disclose.

### Acknowledgements

[STUDENT] contributed to this work in the [FUNDED PROGRAM]

### Data Availability

The CRSS data is publicly available at

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

## 9.

### CRedit authorship contribution statement

**First Author:** Conceptualization, Investigation, Writing - original draft, Visualization. **Second Author:** Supervision, Methodology, Writing - review and editing. **Third Author:** Investigation, Methodology. **Fourth Author:** Data curation, Writing - review and editing.

### References

- Herbert, G., 2019. Crash Report Sampling System: Imputation. Technical Report DOT HS 812 795. National Highway Traffic Safety Administration.
- 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.
- Raghunathan, T., Solenberger, P., Berglund, P., van Hoewyk, J., . Ivedere: Imputation and variation estimation software. URL: <https://www.src.isr.umich.edu/software/iveware/>.