

Highlights

Modeling the Need for an Ambulance based on Automated Crash Reports from Cell Phones

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.
- Explicit Incorporation of Imbalanced Costs
- Perennial Machine Learning Challenge: Imbalanced Datasets

Modeling the Need for an Ambulance based on Automated Crash Reports from Cell Phones

First Author^{a,b}, Second Author^a, Third Author^{a,c} and Fourth Author^c

^aSchool, University,

^bOther School,

^cOther Department, University,

ARTICLE INFO

Keywords:

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

ABSTRACT

New Google Pixel 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 probability p 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 such a tradeoff threshold is ethically problematic, but governments implicitly choose such a tradeoff when they set budgets for emergency services.

We consider and interpret several options for θ , some of which consider a relationship between the total number of FP and FN up to that value of p , and others consider the marginal relationship around that value of p . Once the threshold criteria are chosen, the problem turns to choosing and tuning a model that best satisfies the tradeoff, saving both money and lives.

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 time of day and weather, data the emergency dispatcher has before the notification. The “Medium” level adds the age and sex of the cell phone user and information about the location. The “Hard” level adds information 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

1.1. Outline

- Dataset

– CRSS

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

* 2016-2020, 2021

* Over-represents more serious crashes

- Feature Selection and Engineering
- Discretization
- Imputing Missing Values

- Imbalanced Data

- Can't use SMOTE
- Class Weights
- Focal Loss
- Bagging and Boosting Methods
- Moving the Discrimination Threshold

- Threshold Options

- Choose the precision that is politically acceptable
- Total Precision, including Prior Probability equals Posterior Probability
- Marginal Precision

- Results and Conclusions

2. Literature Review

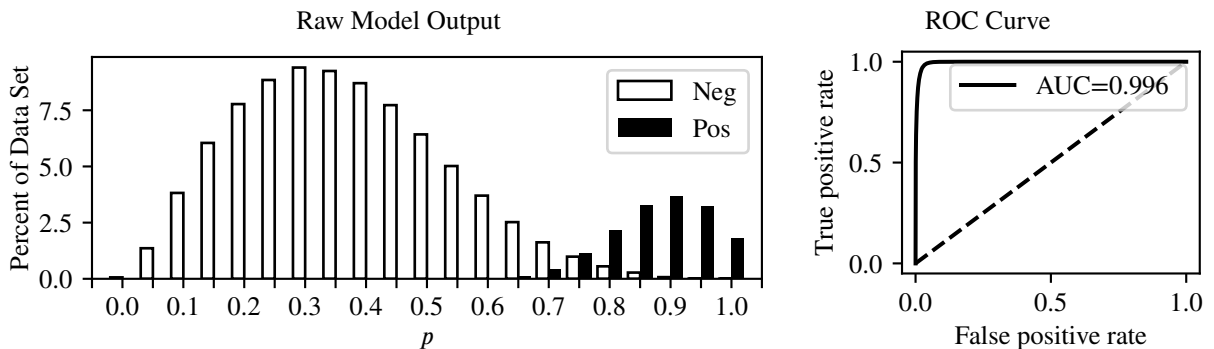
3. Dataset

4. Methods

4.1. Analysis of Results

Our ML algorithms assign to each sample (feature vector, crash person) a probability $p \in [0, 1]$ that the person needs an ambulance. The histogram below left shows the percentage of the dataset in each range of p , showing the percentages for the negative class (“Does not need an ambulance”) and the positive class (“Needs an ambulance”). On the right, the Receiver Operating Characteristic (ROC) curve, and particularly the area under the curve (AUC), is a metric for how well the model separates the two classes, with $AUC = 1.0$ being perfect and $AUC = 0.5$ (the dashed line) being just random assignment with no insight.

We would love to have results like in the graphs below, where the machine learning (ML) algorithm nearly perfectly separates the two classes. There is some overlap between $p = 0.6$ and $p = 0.8$ with some samples the algorithm misclassifies, but the model clearly separates most samples. Having an AUC of 0.996 would be amazing.



Unfortunately, our test results do not look quite that nice. They do not separate the two classes as well. Some distributions are clustered to one side or in the middle. Some models give the results in $p \in [0, 1]$ rounded to two

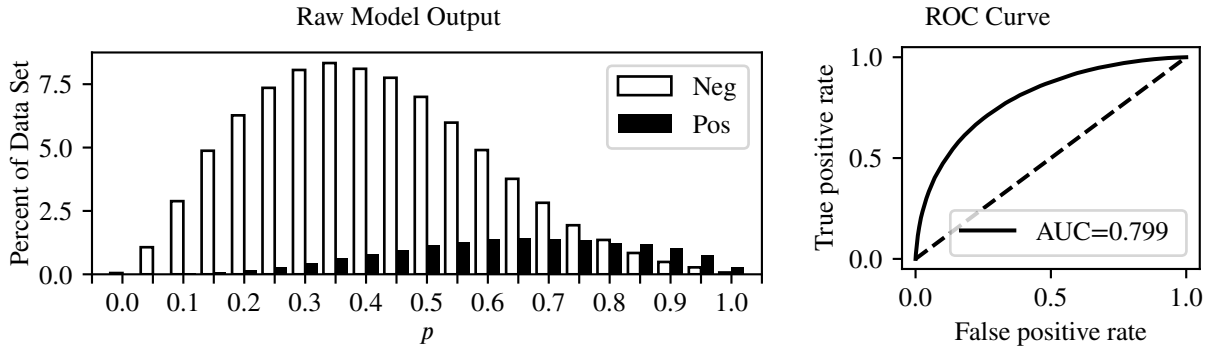
decimal places so that we cannot hope for a level of detail beyond that, and one algorithm, Bagging, gives p rounded to only one decimal place.

Let us look at some examples. In all of them, AUC is in the range [0.7, 0.8], so the various models separate the positive and negative classes about equally well overall, with none being dramatically better or worse. We will later show how we investigated which models do a better job in the ranges of interest.

BRFC_Hard_Tomek_0_alpha_0_5_v1_Test

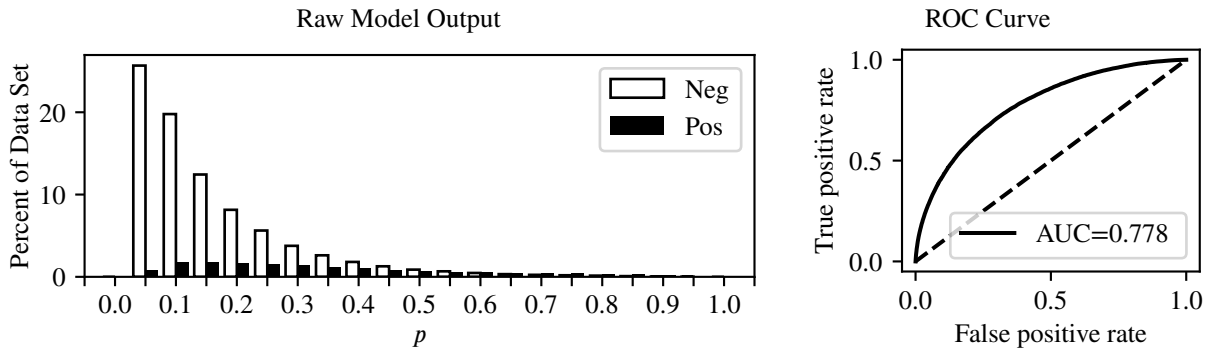
This model does not separate the negative and positive classes as well as the ideal, giving a much lower AUC (area under the ROC curve). These results are actually from the same model as the ideal above, but the ideal are the results on the training set and below on the test set, showing overfitting.

In these results, the 100 most frequent values comprised 93% of the results, meaning that, while there is some noise making the distribution look continuous, it is mostly discrete to two decimal places, so we cannot hope for fine detail in tuning the decision threshold.



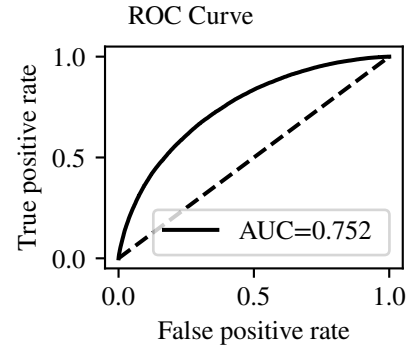
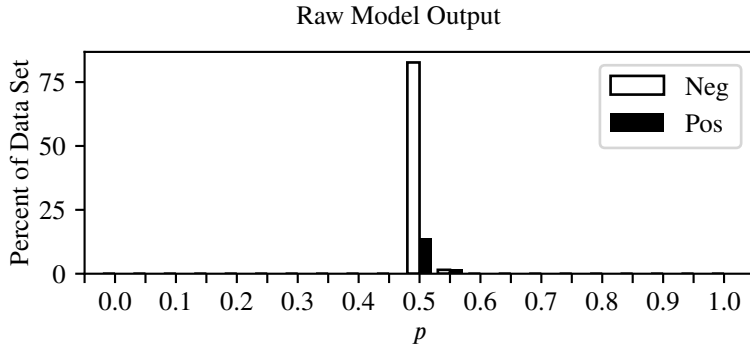
KBFC_Hard_Tomek_0_alpha_0_5_gamma_0_0_v1_Test

This model is almost as effective at separating the two classes (ROC = 0.778), but the distribution is skewed to the left. Its results were nearly continuous, with the 214,070 samples returning 210,157 unique values of p , so we can fine tune the decision threshold.

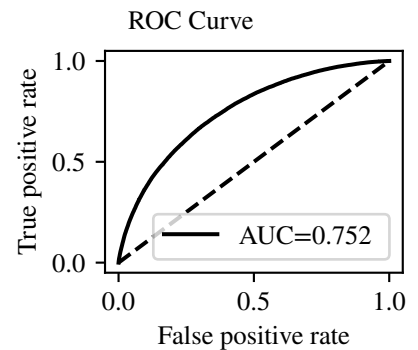
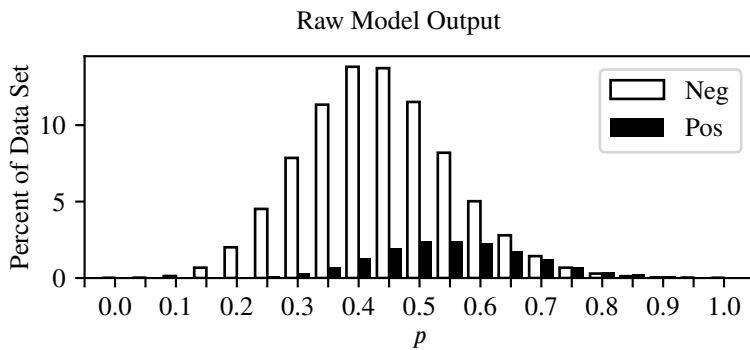


AdaBoost_Hard_Tomek_0_v1_Test

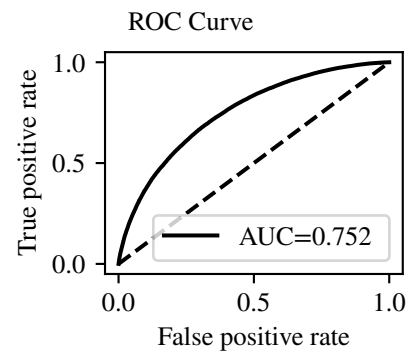
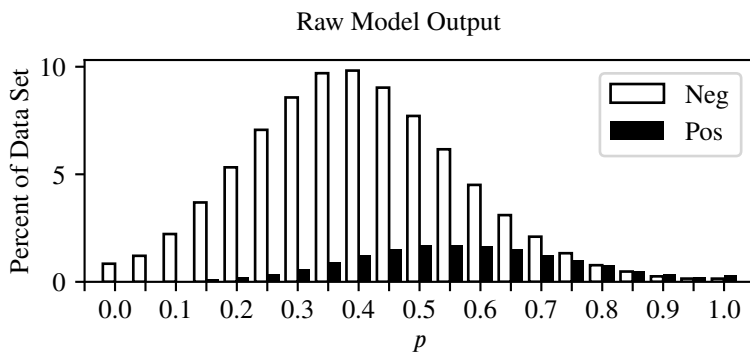
In this model the values are clustered very tightly, but in that small range the 214,070 samples return 210,442 different values of p , so there is much diversity that we can't see in this representation.



To make a useful visualization of the results where we can see the interplay between the negative and positive classes, we can transform the data. A transformation that preserves rank will have no effect on the ROC curve. [Cite] For the graph below, we mapped the smallest value in the set to 0 and the largest to 1.

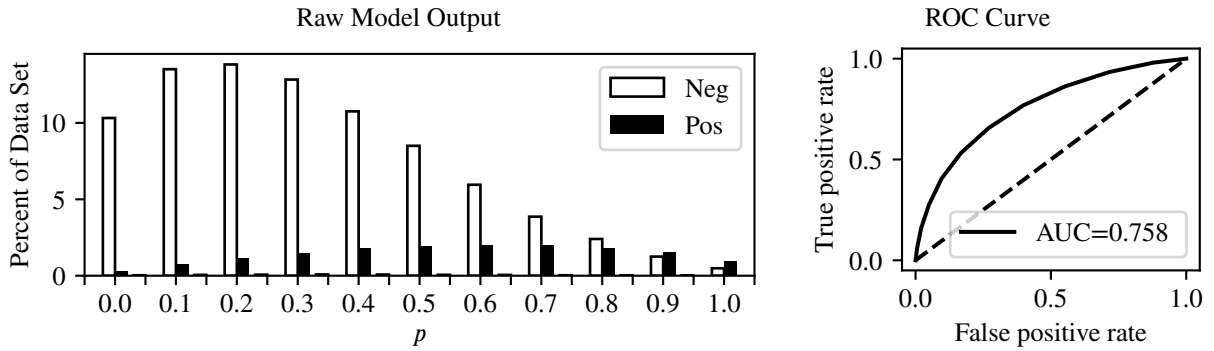


The distribution has long tails, so we can make a more useful visualization by truncating the ends. For this graph we mapped the 0.01 quantile to 0 and the 0.99 quantile to 1 leaving the center 98% of the distribution, and truncated the ends. Our goal in clipping the tails is to make all of the models' results have approximately the same granularity when we choose the decision thresholds that give us the (politically) desired results.



Bagging_Hard_Tomek_0_v1_Test

This model returned 217 different values, but most of them were rare. Taking out the 5% of the data set with the least frequent values, 95% of the samples had only 10 values of p . It may be a useful model, but we will not be able to fine tune the decision threshold.



Other stuff

5. Results

6. Conclusions

7. Discussion

8. Future Work

9. To Do, Notes to Self

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>

10.

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