

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. Emergency dispatchers receive most crash notifications from an eyewitness who can say whether an ambulance is needed, but the automated notification from the cell phone does not provide that information. 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 wait for a call from an eyewitness (perhaps the police) before sending an ambulance. In the “Sometimes” option, the dispatcher relies on an AI 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 model for making a recommendation in the “Sometimes” option. The model gives the probability (based on the available information) that a crash requires an ambulance. If the probability that an ambulance is needed is above some chosen threshold, the method will recommend dispatching an ambulance immediately; below that threshold, the dispatcher should wait to hear from an eyewitness.

We consider the factors in determining the threshold. 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 make such a tradeoff when they set budgets for emergency and medical services. To demonstrate the method in this paper, we have arbitrarily chosen a cutoff of 33%, that there is a 1 in 3 chance that a dispatched ambulance would be needed (TP) and a 2 in 3 chance that it would not (FP). We formulate our marginal ethical tradeoff rate as $\omega = \Delta FP / \Delta TP = 2.0$. We incorporated ω into the model in the class weight and in the decision threshold.

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. Henry’s Questions 23 May 2023

1.1. Abstract

I made most of the changes you suggested for the abstract. (Above)

I changed “police” to “emergency dispatcher” rather than “911 dispatcher” so as to not annoy the people (like the editor) in countries that use 119.

I think the abstract is now too long, but that’s a problem for later.

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

1.2. Threshold, Prior and Posterior Probabilities

Thank you for your summary of the decision problem. I will adapt the language here. I will try to answer your questions through an example. Please let me know if I don't understand it well.

I'm trying to understand the \hat{p} you used. I understand p as the probability that a given sample is in Class 1. Because you related \hat{p} to π_1 , I suspect what \hat{p} means is the percentage of samples the model (with our choice of decision threshold) classifies as being in Class 1. Is that correct? In the big chart below,

$$\hat{p} = \frac{FP + TP}{TN + FP + FN + TP}$$

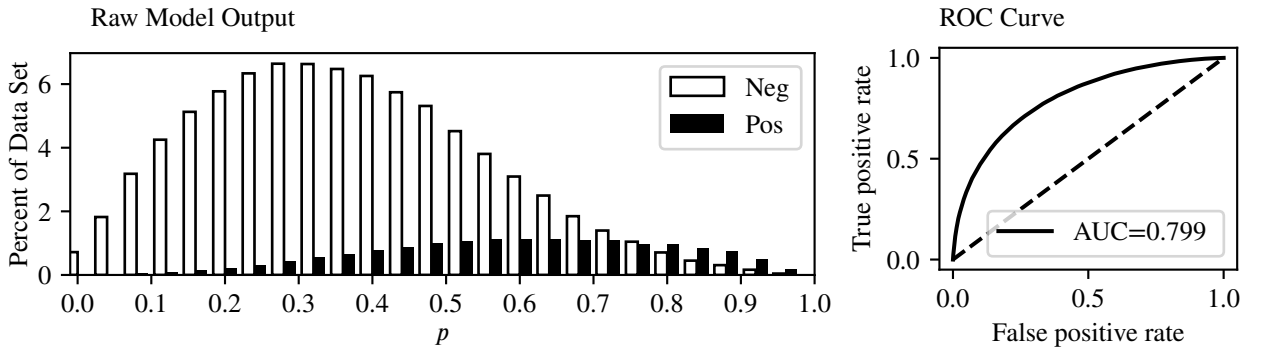
Using the CRSS dataset, our problem has two outcomes.

Class 0	“No ambulance needed”	$\pi_0 \approx 85\%$
Class 1	“Ambulance needed”	$\pi_1 \approx 15\%$

We will run several models, but a particular model gives, for each input vector (sample), a probability p that the sample is in Class 1. We want to pick a discrimination threshold (decision threshold) θ such that if for a particular crash notification $p > \theta$, then our recommendation system recommends that the emergency dispatcher send an ambulance.

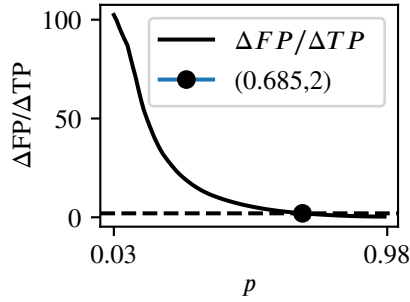
Below is the test results of one of our best models, the Balanced Random Forest Classifier with no class weights ($\alpha = 0.5$). The histogram gives, for each range of p , the percent of the total dataset with Class 0 (Neg) and Class 1 (in that range of p).

Until we choose the discrimination threshold θ , we can't have a confusion matrix. Once we choose θ , then all of the negative samples with $p < \theta$ are true negatives, and all of the negative samples with $p > \theta$ are false positives; conversely for the positive samples.



In the table below,

- The first three columns are the same information as in the histogram. “Neg” (“Pos”) is the number of negative (positive) samples whose probability of being in Class 1 is in that range of p .
- The fourth column, “Neg/Pos,” is the number of unneeded ambulances sent for each needed ambulance sent, for samples with p in that range. This is the marginal cost of having θ below this value of p . These values are given in this plot, emphasizing where $\text{Neg/Pos} = \Delta FP / \Delta TP = 2$.



The “TN,” “FP,” “FN,” and “TP” columns are the confusion matrix if we choose θ in that range of p .

- The seventh column, “FP/TP” is related, but it’s the total cost in terms of how many unneeded ambulances we’d send for each one we needed if θ were that value of p .
- The last three columns are Precision, Recall, and \hat{p} (If I understand \hat{p} correctly).

p	Neg	Pos	Neg/Pos	TN	FP	FN	TP	FP/TP	Prec.	Rec.	\hat{p}
0.00	107	0	inf	107	180,138	0	33,825	5.33	0.16	1.00	1.00
0.05	2,287	20	114.35	2,394	177,851	20	33,805	5.26	0.16	1.00	0.99
0.10	6,177	62	99.63	8,571	171,674	82	33,743	5.09	0.16	1.00	0.96
0.15	10,435	188	55.51	19,006	161,239	270	33,555	4.81	0.17	0.99	0.91
0.20	13,424	388	34.60	32,430	147,815	658	33,167	4.46	0.18	0.98	0.85
0.25	15,746	619	25.44	48,176	132,069	1,277	32,548	4.06	0.20	0.96	0.77
0.30	17,256	923	18.70	65,432	114,813	2,200	31,625	3.63	0.22	0.93	0.68
0.35	17,841	1,320	13.52	83,273	96,972	3,520	30,305	3.20	0.24	0.90	0.59
0.40	17,355	1,690	10.27	100,628	79,617	5,210	28,615	2.78	0.26	0.85	0.51
0.45	16,597	2,042	8.13	117,225	63,020	7,252	26,573	2.37	0.30	0.79	0.42
0.50	14,984	2,470	6.07	132,209	48,036	9,722	24,103	1.99	0.33	0.71	0.34
0.55	12,810	2,725	4.70	145,019	35,226	12,447	21,378	1.65	0.38	0.63	0.26
0.60	10,493	2,972	3.53	155,512	24,733	15,419	18,406	1.34	0.43	0.54	0.20
0.65	8,062	3,037	2.65	163,574	16,671	18,456	15,369	1.08	0.48	0.45	0.15
0.70	6,040	2,953	2.05	169,614	10,631	21,409	12,416	0.86	0.54	0.37	0.11
0.75	4,144	2,893	1.43	173,758	6,487	24,302	9,523	0.68	0.59	0.28	0.07
0.80	2,902	2,627	1.10	176,660	3,585	26,929	6,896	0.52	0.66	0.20	0.05
0.85	1,801	2,530	0.71	178,461	1,784	29,459	4,366	0.41	0.71	0.13	0.03
0.90	1,043	2,166	0.48	179,504	741	31,625	2,200	0.34	0.75	0.07	0.01
0.95	587	1,597	0.37	180,091	154	33,222	603	0.26	0.80	0.02	0.00
1.00	154	603	0.26	180,245	0	33,825	0	nan	nan	0.00	0.00

Here I’ve zoomed in on three smaller ranges. Interestingly, it’s not useful to zoom in further, because most values of p given by this implementation of the classifier are only given to two digits.

	Neg	Pos	Neg/Pos	TN	FP	FN	TP	FP/TP	Prec.	Rec.	\hat{p}
p											
0.48	2,990	453	6.60	126,315	53,930	8,688	25,137	2.15	0.32	0.74	0.37
0.49	3,020	533	5.67	129,335	50,910	9,221	24,604	2.07	0.33	0.73	0.35
0.50	2,874	501	5.74	132,209	48,036	9,722	24,103	1.99	0.33	0.71	0.34
0.51	2,804	533	5.26	135,013	45,232	10,255	23,570	1.92	0.34	0.70	0.32
0.64	1,582	586	2.70	162,135	18,110	17,838	15,987	1.13	0.47	0.47	0.16
0.65	1,439	618	2.33	163,574	16,671	18,456	15,369	1.08	0.48	0.45	0.15
0.66	1,376	561	2.45	164,950	15,295	19,017	14,808	1.03	0.49	0.44	0.14
0.67	1,288	637	2.02	166,238	14,007	19,654	14,171	0.99	0.50	0.42	0.13
0.68	1,241	554	2.24	167,479	12,766	20,208	13,617	0.94	0.52	0.40	0.12
0.69	1,082	631	1.71	168,561	11,684	20,839	12,986	0.90	0.53	0.38	0.12
0.70	1,053	570	1.85	169,614	10,631	21,409	12,416	0.86	0.54	0.37	0.11
0.71	922	587	1.57	170,536	9,709	21,996	11,829	0.82	0.55	0.35	0.10
0.76	664	558	1.19	174,422	5,823	24,860	8,965	0.65	0.61	0.27	0.07
0.77	627	524	1.20	175,049	5,196	25,384	8,441	0.62	0.62	0.25	0.06
0.78	585	532	1.10	175,634	4,611	25,916	7,909	0.58	0.63	0.23	0.06
0.79	568	529	1.07	176,202	4,043	26,445	7,380	0.55	0.65	0.22	0.05
0.80	458	484	0.95	176,660	3,585	26,929	6,896	0.52	0.66	0.20	0.05
0.81	429	514	0.83	177,089	3,156	27,443	6,382	0.49	0.67	0.19	0.04
0.82	399	535	0.75	177,488	2,757	27,978	5,847	0.47	0.68	0.17	0.04

How shall we choose θ , the discrimination threshold?

For example, if we choose the default $\theta = 0.50$, we would send ambulances to 34% of the automatically reported crashes. The total cost is 1.99 unneeded ambulances per needed ambulance. The marginal cost, the difference between making $\theta = 0.50$ and making $\theta = 0.51$, is over 5 (5.74) unneeded ambulances per needed ambulance. We would be sending an ambulance to each crash with at least a $1/(5.74 + 1) \approx 15\%$ chance of needing an ambulance.

One goal of my analysis is figuring out how to choose θ given some marginal probability that an ambulance is needed, given explicitly or implicitly by the people funding the emergency services. I think this is actually how the decision is likely to be made by the politicians: We're willing to send an ambulance early (before an eyewitness report) if there is some probability that it's needed.

The option I'm exploring is sending an ambulance when there's at least a 33% chance it will be needed, which happens when Neg/Pos = 2, at about $\theta = 0.68$. The total cost would be 0.99 unneeded ambulance, and we would be sending an ambulance to 13.16% of the crashes.

If we wanted there to be at least a 50% chance that an ambulance is needed, then we would choose $\theta = 0.80$, where Neg/Pos = 1.

If we wanted $\hat{p} \approx \pi_1$, then we would choose $\theta = 0.65$.

Does that decision-making method make sense?

1.3. Other Questions: Dataset

The CRSS dataset intentionally over represents more serious crashes; in all police-reported crashes Class 1 is much smaller (2-3%), but it is also true that very minor crashes (parking lot fender benders) have a similar deceleration profile to hard braking, so minor crashes are less likely to be detected by the phone. Also, the automated report only goes to the emergency dispatcher if the phone's owner does not respond promptly to the phone. So we are going to wave our hands and say that the CRSS data is the best approximation we have to the set of crashes reported by automated cell phone reports. Does that approach seem reasonable?

1.4. Other Questions: Validation Set

I've done my work so far with the data split 70/30 into training and test, and then I did that twice, splitting with a different random seed, to compare results and see whether the differences were within randomness. Should I do a 60/20/20 training/validation/test set, or use 5-fold cross-validation on the training set? Or does it matter?

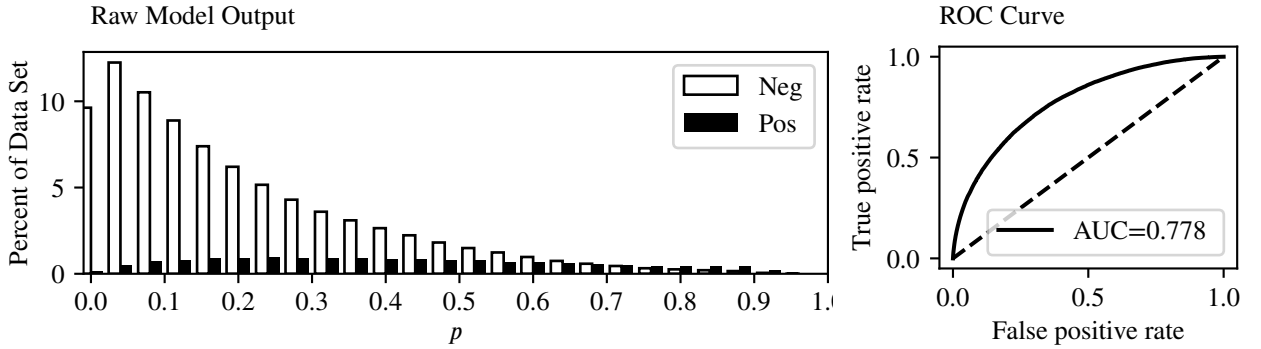
1.5. Other Questions: Class Weights

Below are the raw model outputs y_{proba} for the (neural network) Keras Binary Crossentropy Classifier with three different class weights α .

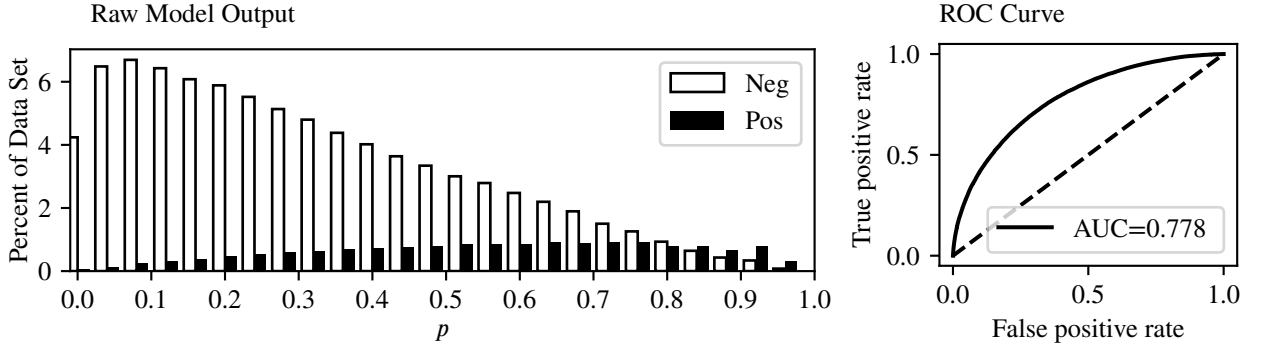
I had thought the point of class weights was that class weights would put more weight on the misclassified elements of the positive class and make the algorithm would do a better job of separating the two classes. Using the area under the ROC curve (AUC) as a measure of how well the model separates the classes, the difference between these three models with different class weights is within randomness. It seems that raising the class weight just pushes both classes together to the right with no useful effect. Do class weights basically have the same effect as shifting the decision threshold θ ?

On the next page I show that if you linearly transform the p values so that $p = 0.5$ where $\Delta FP / \Delta TP = 2.0$, then you get nearly the same confusion matrix. The AUC is invariant under the transformation.

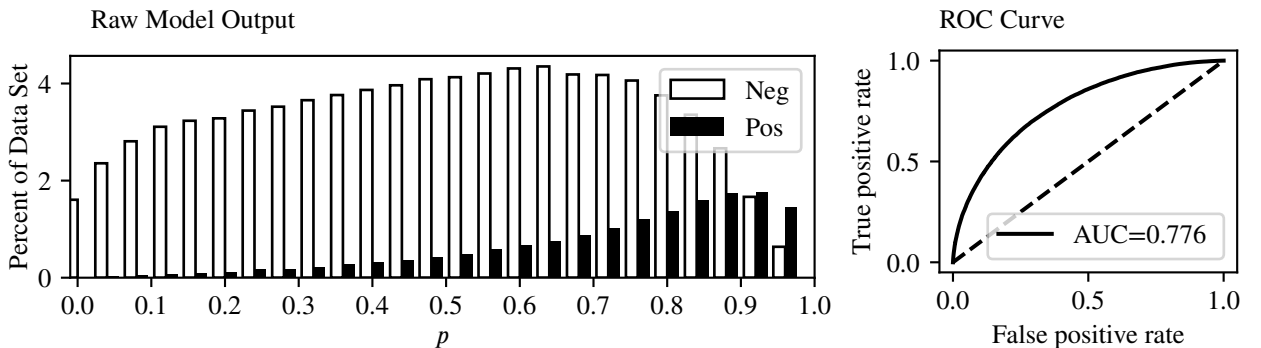
Model 1: $\alpha = 0.5$ for no class weights



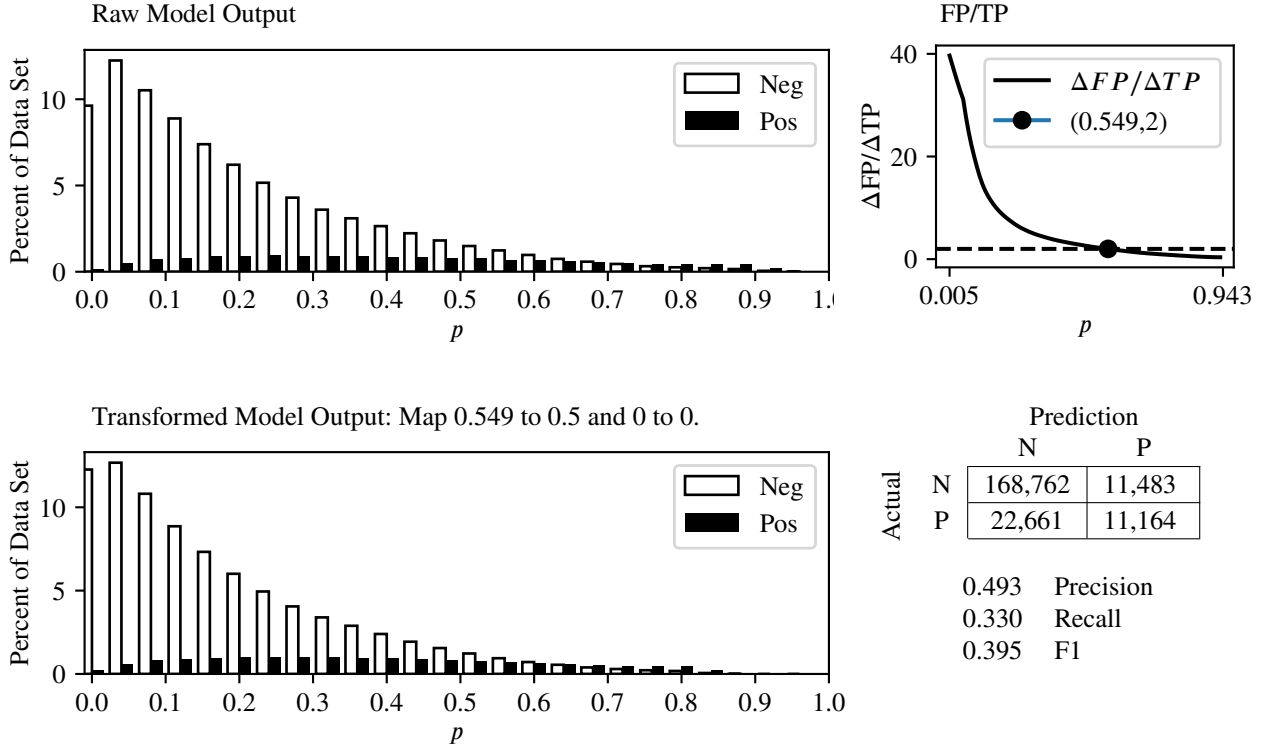
Model 2: $\alpha = 0.67$ for 33% chance the ambulance is needed



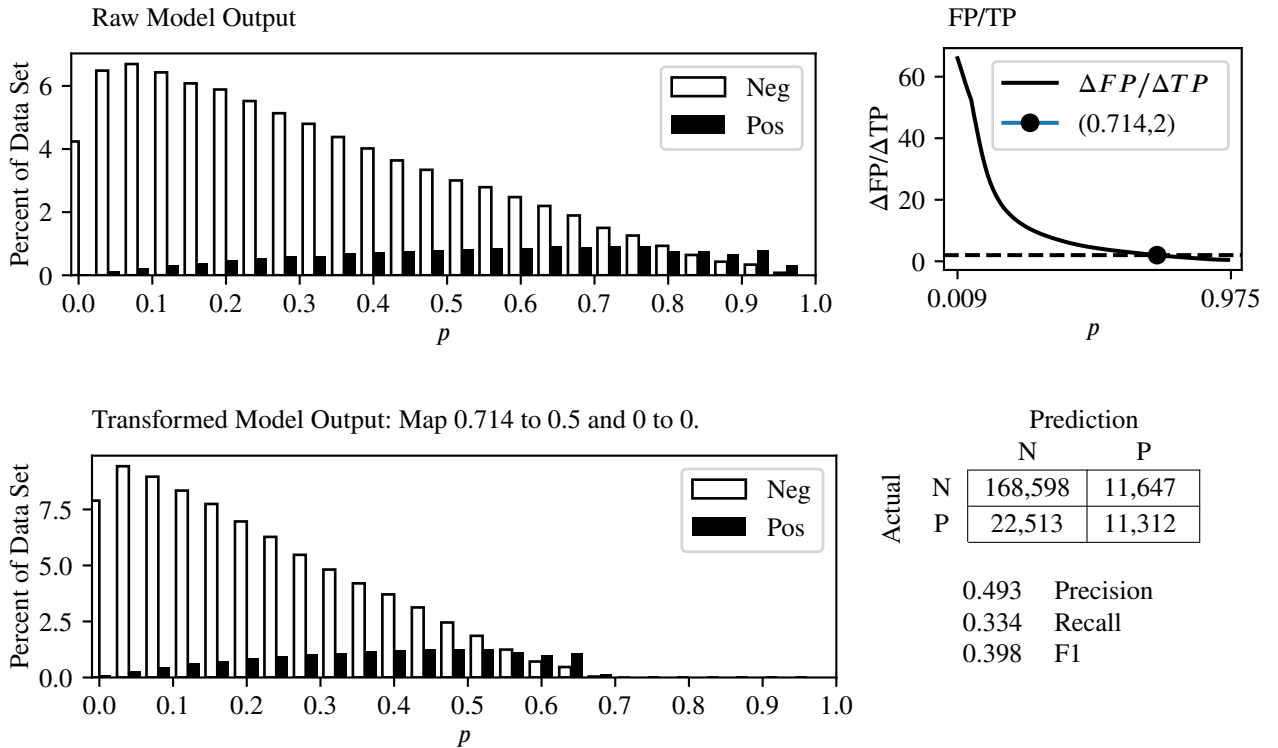
Model 3: $\alpha = \pi_0 = 0.84$ for class balance



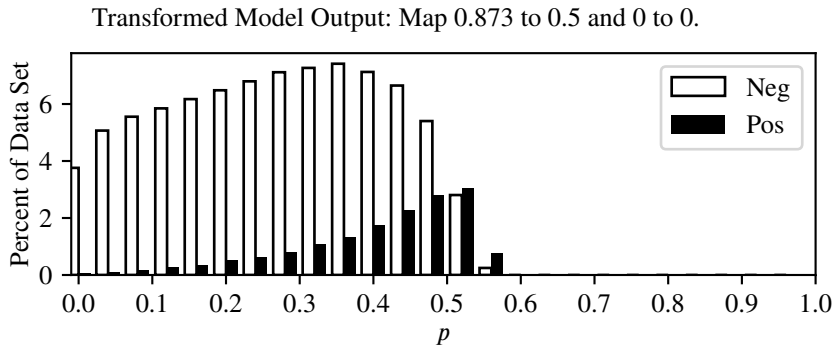
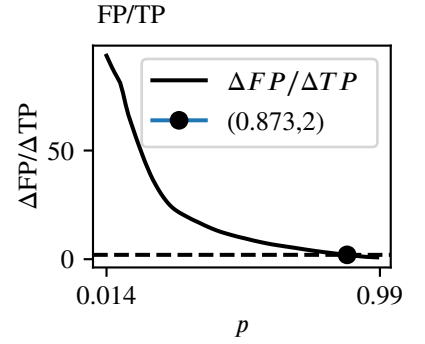
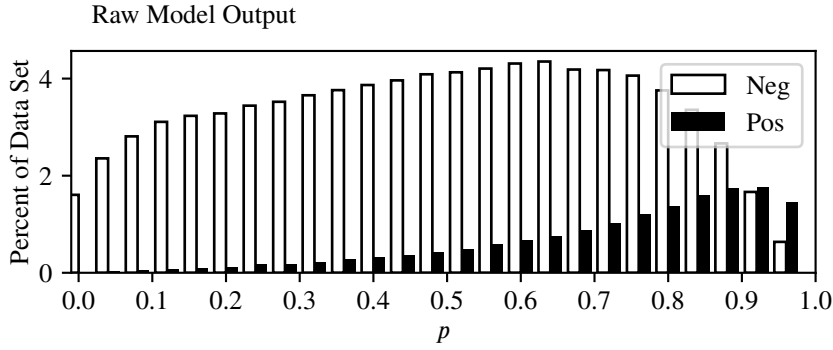
Model 1: $\alpha = 0.5$ for no class weights



Model 2: $\alpha = 0.67$ for 33% chance an ambulance is needed



Model 3: $\alpha = \pi_0 = 0.84$ for class balance



		Prediction	
		N	P
Actual	N	168,381	11,864
	P	22,587	11,238

0.487 Precision
0.332 Recall
0.395 F1

I ran the same Keras three with the same three class weights, but using $\Delta FP / \Delta TP = 1$, and got similar results, that when you “normalize” the output to center where $\Delta FP / \Delta TP = 1$, you get basically the same results. One had better recall (0.146 instead of 0.156), but otherwise the same. Then I tried it with $\Delta FP / \Delta TP = 3$, and also basically the same.

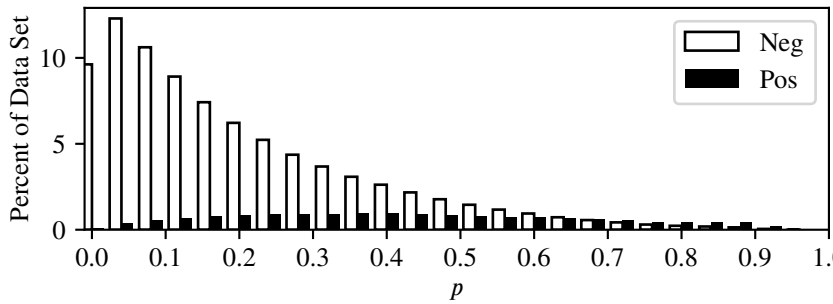
Any differences may be attributable to the values of p not really being continuous, as most of them only have two decimal places, like 0.52, with very few like 0.5246, so the differences in the metrics may be attributable to rounding errors.

1.6. Other Questions: Overfitting?

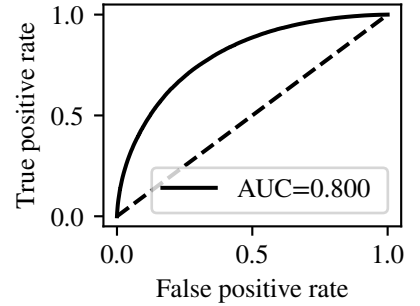
When I use the Keras Binary Crossentropy Classifier and test for overfitting by running the classifier on both the training and test sets, I get basically the same thing, which (I think) means it's not overfitting.

```
y_proba = estimator.predict_proba(X_train)
```

Raw Model Output on Training Set

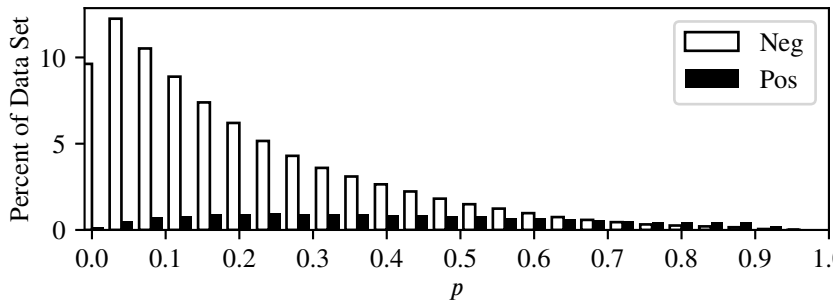


ROC Curve

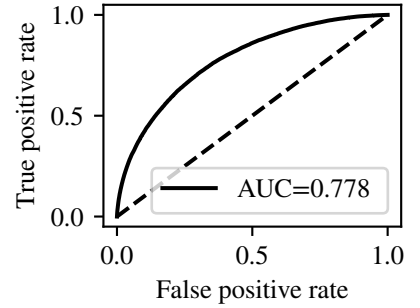


```
y_proba = estimator.predict_proba(X_test)
```

Raw Model Output on Test Set



ROC Curve

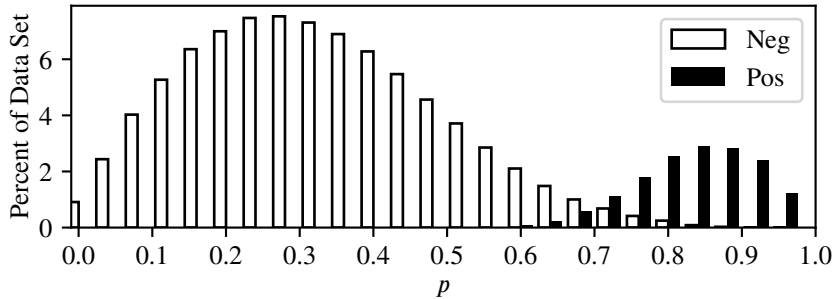


[continued on next page]

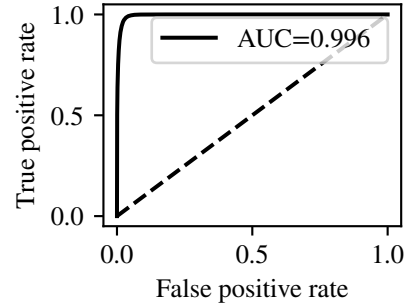
When I use the Balanced Random Forest Classifier, however, I get really different results, which is a sign that it's overfitting.

```
y_proba = estimator.predict_proba(X_train)
```

Raw Model Output on Training Set

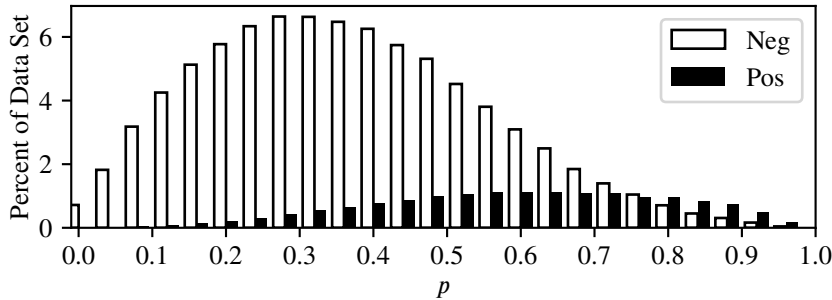


ROC Curve

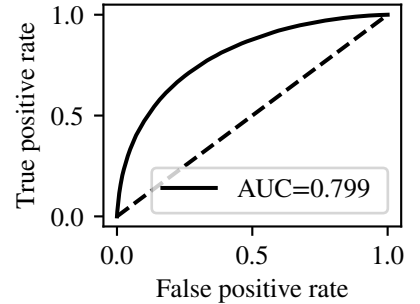


```
y_proba = estimator.predict_proba(X_test)
```

Raw Model Output on Test Set



ROC Curve



I tried several of the usual tricks to get it to not overfit, like changing the number of trees, the maximum depth, and the maximum number of leaf nodes, and they gave me poorer results for both the training set and the test set, so I don't know that that's better. The Balanced Random Forest Classifier is by far the best model algorithm I've used for giving the best AUC, precision, accuracy, and F1. Is overfitting a problem if it gives the best results on the test set?

2. Introduction

3. Literature Review

4. Dataset

5. Methods

6. Results

7. Conclusions

8. Discussion

9. Future Work

10. 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>

11.

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