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[a,b], Second Author[a], Third Author[a,c] and Fourth Author[c]

[a]*School, University,*
[b]*Other School,*
[c]*Other Department, University,*

ARTICLE INFO

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. Such a model will not be perfect, with many false positives (sending an ambulance when one is not needed) and some false negatives (not sending an ambulance when one is needed), but better than random. How much better depends on several things that we will investigate.

A key idea underlying this analysis is that 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. We propose a way to interpret class weights as the ethical rate of tradeoff.

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. Whether a model is "good" is partly a political question, weighing prompt medical care against its high cost, but given a parameter p of how to weigh those costs, we can build that tradeoff into the model.

We used the data of the Crash Report Sampling System (CRSS).This data 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

2.1. Apps to Detect Crashes

The idea of using the accelerometer in a cell phone to detect a crash and notify the police goes back to at least 2011 with *WreckWatch*, an app prototype by White, Thompson, Turner, Dougherty and Schmidt (2011). A few years later, Aloul, Zualkernan, Abu-Salma, Al-Ali and Al-Merri (2014) proposed an app that would detect not only a crash but its severity, and send the phone owner's medical information. In September 2019, Google hinted that it would have crash detection in its Pixel phones soon (Rahman, 2019), and it is now available (Google). In November 2021, the Wall Street Journal announced that Apple was thinking of introducing such an app in its iPhones and Apple watches in 2022 Winkler (2021), but as of this writing, such an app had not come out yet. The App Store does have some unverified third-party apps, like those from Sosmart SAP and BlinkApp LLC.

2.2. Imbalanced Data

In building a model to predict, based on records of roadside inspections, traffic violations, and previous crashes, future crashes for trucks, Lack, Berkow and Gao (2021) described the imbalanced data problem well.

> Initial models "correctly" classified no-crash versus crash instances 99% of the time, but almost never correctly predicting a crash—a major failure in achieving the goal of this analysis and a common issue in unbalanced datasets.

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

Our work in this paper uses the most straightforward of machine learning (ML) algorithms, the binary classifier. We want to answer, "Should we send an ambulance to this crash," and are using historical data answering for each crash, "Did we use an ambulance?" We want to build a model that will look at the data we have for a crash and return a prediction. To build it, we will separate our data into a training set and a test set; use the training set to build the model, then evaluate the model on the (unseen during model building) test set.

The model will not be perfect. We want to send ambulances to all of the crashes that need one (True Positive), and not send ambulances to crashes that don't (True Negative). Sending an ambulance when one is not needed (False Positive) is unnecessarily expensive, but if we don't send an ambulance when one is needed (False Negative), someone might die unnecessarily.

We can visually organize the success and failures of our binary classification model in a *confusion matrix* (also called *error matrix*, or *contingency table*).

		Prediction	
		N	P
Actual	N	TN	FP
	P	FN	TP

Most machine learning algorithms are designed to maximize *accuracy*, the proportion of classifications that were successful.

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$$

In many applications, the accuracy is straightforward and useful, but not if the data are imbalanced. In our case, in all of the reported crashes in Louisiana in 2014-2018, there were 645,748 people involved in the crashes, and 55,164 used an ambulance (`OCC_MED_TRANS_CD = A`), 8.5% of the total.

If we built a model that predicted that none of the crashes required an ambulance, our model would have 91.5% accuracy.

		Prediction	
		N	P
Actual	N	590,584	0
	P	55,164	0

$$\text{Accuracy} = \frac{590{,}584 + 0}{590{,}584 + 0 + 55{,}164 + 0} \approx 0.915$$

In many applications, 91% accuracy would be good. Why is it different in our context? For two reasons. First, our model is useless because we have entirely misclassified all of the positive samples. Second, accuracy gives the same weight to false positives and false negatives, but in our application, their costs are very different. The cost of a false positive (sending an ambulance when one is not necessary) is measured in money, while the cost of a false negative (not sending an ambulance promptly when one is necessary) is measured in lives. We are willing to trade off sending *p* number of unnecessary ambulances if it means that we will send one more necessary ambulance. [Choosing the value of *p* is a question for ethicists and politicians that we will discuss later.]

There are four [citation] categories of methods for building a machine learning model on an imbalanced dataset, and we will use all four in this paper.

- Data-level techniques to artificially balance the dataset

- Modifying the loss function in building the model

- Modifying the model-building algorithm

- Changing the metric in evaluating the model

## 2.3. Data-Level Techniques for Imbalanced Data

Consider this two-dimensional training dataset, which we will use to illustrate data-level techniques for handling imbalanced datasets. In real problems, of course, the dataset could have a hundred dimensions and a million samples. 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").
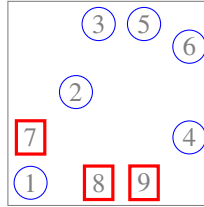
Many algorithms, and variations thereon, have been proposed to balance the two classes before applying a machine learning algorithm to build a model to classify new samples as positive or negative. WARNING: Vast oversimplification ahead. Our goal here is to give the general idea of each method.

2.3.1. Random Undersampling

Random undersampling balances the two classes by randomly deleting elements of the majority class until the two are balanced. The major drawback of this method is that you throw away information about the majority class. If the majority class is many more times the size of the minority, you lose almost all of the data.
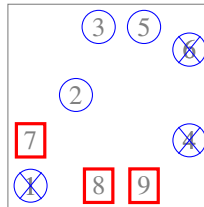

2.3.2. Random Oversampling

Random oversampling creates duplicates of minority class samples until the sets are balanced. This method has a similar effect to using class weights, introduced below.
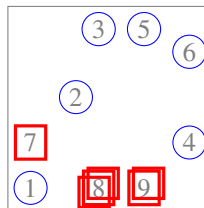
2.3.3. Synthetic Minority Sampling TEchnique (SMOTE)

SMOTE (Chawla, Bowyer, Hall and Kegelmeyer, 2002) is one of the most popular oversampling methods for balancing a dataset with continuous numerical data. It creates new synthetic minority samples "between" original minority samples, not necessarily at the midpoint by choosing a number in $(0, 1)$, multiplying the difference (in each dimension) from point A to B by that constant, and adding it to A.

In the diagram, the solid red squares represent new synthetic samples between pairs of original minority-class samples. SMOTE does not consider the positions of the majority-class samples, only considering the difference in number of nodes to bring the two classes closer to parity.
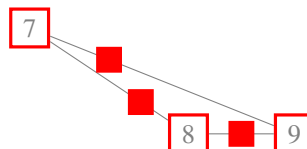
One challenge with SMOTE is that it is only useful for datasets with continuous numerical data, and our data is almost all categorical. What is between "car" and "school bus," or between "parking lot" and "highway"? SMOTE has a variant, SMOTE-NC (Nominal and Continuous) that can handle datasets with some nominal (categorical) features, but most of the features need to be continuous; thus, we will not be able to use SMOTE or similar techniques for our work.

2.3.4. Tomek's Links

Tomek (1976) proposed a method of undersampling that assumes that the majority and minority classes should (at least locally) be clustered. If an sample A of the majority class and a sample B of the minority class are each other's nearest neighbors, then one of them is not clustered with its own class. Since we are trying to undersample the majority class, assume that the element of the majority class is noise (or an error, or just not useful), and delete it.

In the diagram below, samples #1 and #7 are Tomek links, because they are each other's nearest neighbors and of different classes. Samples #4 and #9 are not Tomek links, because while 9 is 4's nearest neighbor, 9's nearest neighbor is 8, not 4.

In the context of modeling crash severity from police reports, why would sample #1 not need an ambulance when its characteristics are so close to those of #7 and not near most of the other crashes without serious injury? The reason could be errors in the records, or luck/providence/fate. It could also be that the difference between property damage only and serious injury is influenced by thousands of variables we cannot measure or know, all of the physics of crash forces acting on the bones and structures of the human body. The best we can say is that the outcome in #1 cannot be predicted by the information that we have, so that sample will not help in constructing a model based on the available data; therefore, we can reasonably delete it from the training set.

Tomek's Links can also be run iteratively. Sample #7 had #1 as its nearest neighbor, but once #1 is deleted, then #2 and #7 are each other's nearest neighbors of different classes, thus are Tomek links, and we can delete #2.
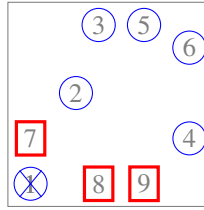
2.3.5. Oversampling Image Data

Extracting knowledge from a database of tabular numerical or categorical data is difficult, but a database of images is a challenge of a different magnitude. An imbalanced labeled image dataset for crash prediction modeling might be a thousand images taken ten seconds before a crash and a million images taken ten seconds before ... nothing happened. Deep neural networks (DNN) and (deep) convolutional neural networks (DCNN and CNN) are common methods for image data. ? introduced Generative Adversarial Networks, which can be used to generate synthetic samples to balance the dataset. Given the power of the tools for image recognition, many researchers make non-image data look (to the computer) like images to take advantage of the tools.

2.4. Modifying the Loss Function

Machine learning algorithms generally work iteratively by picking a starting point for the constants in the model (often a random guess), measuring the error, making a small perturbation in the model constants, measuring the new error in the candidate model, and comparing the two. If the new error is less, use the candidate model; if not, go back to the old one. Repeat.

A common way to measure the error in binary classification is log loss (binary cross-entropy loss, logistic loss). For each sample in the training set we have the answer to the question (the *label*), 0 for "no ambulance," 1 for "ambulance," and the candidate model gives a probability $p \in (0, 1)$ that this sample will need an ambulance. The log loss is the sum over the samples of the log of the error. If the sample is in the majority class, the true value is 0, and if the model gives a value of p, then $\log(1 - p)$ gives $\log(1) = 0$ if the model is perfectly correct and $\log(0) = -\infty$ if the model is perfectly wrong on this sample. Similarly, for a sample in the minority class, $\log(p)$ gives $\log(1) = 0$ if the model correctly classifies the sample. If y is the label, then the log loss for each sample is (Pedregosa, Varoquaux, Gramfort,

Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay, 2011)

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

Note that if $y = 0$, then the first term is 0 for any value of p, so only the second term (majority class) is relevant; correspondingly, if $y = 1$, then only the first term is relevant, so a clearer expression might be

$$L(y, p) = -\left(\left[\text{Loss if minority class } (y = 1)\right] + \left[\text{Loss if majority class } (y = 0)\right]\right)$$

Since the logs of $p \in (0, 1)$ will be negative, the negative in front makes the loss positive, and the iterations of the algorithm will seek to minimize it.

2.4.1. Class Weights (Sample Weights, Cost Sensitive Analysis)

Class weights change the error metric, giving more weight to misclassification of minority class samples (King and Zeng, 2001). Giving double weight to misclassified minority class samples would have the same effect as duplicating all of the negative class samples in oversampling. To achieve balance in the contribution of the two classes to the loss function,

$$\text{Let } r = \frac{\text{Total number of samples}}{\text{Number of minority samples}} \qquad \text{Let } \alpha = \frac{r}{r + 1} \qquad 1 - \alpha = \frac{1}{r + 1}$$

$$L(y, p, \alpha) = -(\alpha y \log(p) + (1 - \alpha)(1 - y) \log(1 - p))$$

2.4.2. Focal Loss

This method is recent, but has appeared in the crash analysis (see Yu, Wang, Zou and Wang (2020) referenced below). Focal loss (Lin, Goyal, Girshick, He and Dollár, 2017) adds another factor to the loss function that gives more weight to samples that are badly misclassified, and less weight to samples that are slightly misclassified.

$$L(y, p, \alpha, \gamma) = -\left(\alpha y \log(p)(1 - p)^{\gamma} + (1 - \alpha)(1 - y) \log(1 - p) p^{\gamma}\right)$$

The paper by Lin et al. tested values of $\gamma \in [0.0, 5.0]$, and found that $\gamma = 2.0$ gives good balance, but the best value depends on the dataset and the goals. Yu et al's paper using focal loss for real-time crash prediction allowed the γ for minority and majority classes to have different values.

$$L(y, p, \alpha, \gamma_1, \gamma_2) = -\left(\alpha y \log(p)(1 - p)^{\gamma_1} + (1 - \alpha)(1 - y) \log(1 - p) p^{\gamma_2}\right)$$

2.4.3. Comparison of Loss Functions

The table below compares the three loss functions. For α, we will assume that the minority class is 10% of the dataset, so $r = 10 \rightarrow \alpha = r/(r + 1) = 10/11 \approx 0.9090$. For focal loss, we will use $\gamma = 2$.

Machine learning algorithms use the loss function when comparing two candidate models, only asking which one is less, with no concern for the actual magnitude. For this reason, the choice of base for the logarithm is inconsequential (we arbitrarily choose base 10 for the chart below); also, that the raw focal loss values are each less than the raw class weights, which are each less than the raw log loss, is not relevant, so we have included normalized values for comparing the three loss functions.

Class	y	p	Raw			Normalized		
			Log Loss	Class Weights	Focal Loss	Log Loss	Class Weights	Focal Loss
Minority	1	0.9	0.04576	0.04160	0.00042	0.04560	0.08292	0.00464
	1	0.7	0.15490	0.14082	0.01267	0.15438	0.28069	0.14136
	1	0.5	0.30103	0.27366	0.06842	0.30002	0.54548	0.76309
Majority	0	0.5	0.30103	0.02737	0.00684	0.30002	0.05455	0.07631
	0	0.3	0.15490	0.01408	0.00127	0.15438	0.02807	0.01414
	0	0.1	0.04576	0.00416	0.00004	0.04560	0.00829	0.00046

Note that, with focal loss, most of the total loss comes from the one badly misclassified minority sample, so to minimize the loss, the algorithm needs to do a better job classifying that sample.

2.5. Algorithm-Level Methods for Imbalanced Data

2.5.1. Bagging

"Bagging" is short for Bootstrapped Aggregating, a variation on random undersampling (Breiman, 1996). In general, bagging takes many random subsets (with replacement) of the samples, run the classifier on each subset, then aggregate the results. In imbalanced data applications, each subset of the samples is all of the n minority samples and n randomly chosen majority samples.

Balanced Random Forest [need citation] is a form of bagging.

In our example, bagging would make a subset of the data with the three minority-class samples (#7, 8, and 9), and three randomly chosen from the majority-class samples, run the classifier; repeat some number of times. Use an ensemble classifier to merge the results.
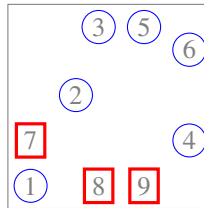
2.5.2. Boosting

Boosting is an iterative method that runs the classifier multiple times. At the end of each iteration, it determines which samples would be misclassified under the current model. In the next iteration, the classifier gives higher weight to the misclassified samples, improving the model on marginal cases. While boosting is not just for imbalanced data, the challenge in imbalanced data is that the minority class samples get misclassified, so boosting would help. A popular implementation is AdaBoost, introduced by Freund and Schapire (1997).

2.6. Using Different Metrics to Evaluate a Model built on Imbalanced Data

Our goal in building a model is to make an informed guess about a future situation, in our case data from an automated call from the phone of a person involved in a crash. Does the person need an ambulance? We don't know for certain, but using past examples we want a model that will make an informed guess.

To build a model (in supervised learning), we have a set of data (historical data in our case) for which we know the answer to the question, "Did this crash person need an ambulance?" We split the data into two parts, training and test, use the training data to build a model, then evaluate the quality of the model using the test data, for which we now have both the actual answer to the question and the model's educated guess.

When building a model, we test many different models with different algorithms and hyperparameters (user-set parameters, like the γ in Focal Loss) and see which model best solves our problem. To compare two models, we need a metric.

When testing a candidate model, for each sample in the test set, the model returns a probability $Pr \in (0, 1)$ that the sample is a member of the positive (minority) class, in our case, the probability that the crash person needs an

ambulance. We then use a threshold (0.5 by default) to classify the prediction as negative (N, 0) if the probability is less than the threshold, and positive (P, 1) if more. Then we compare the prediction with the actual classification (ground truth), which we can organize into a confusion matrix (truth table).

		Prediction	
		N	P
Actual	N	TN	FP
	P	FN	TP

The most obvious metric is *accuracy*, the proportion of classifications that were successful.

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$$

Because accuracy gives the same weight to the two classes, it may not be the best metric for imbalanced data sets (because it may overfit the majority class), nor for situations where the cost of a false positive (sending an ambulance when one is not needed) is different from the cost of a false negative (not sending an ambulance when one is needed). Both situations apply to our problem, so we need to consider different metrics, some of which are affected by data imbalance, and some not.

2.6.1. Recall

Recall, or the True Positive Rate (TPR), in our context is, of the calls that actually needed an ambulance, what proportion did the model classify correctly? Increasing recall would save lives. We really want this number to be high, and are willing to trade off other errors to achieve that goal.

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}}$$

Recall only considers the positive (minority) class of the dataset, so it is unaffected by class imbalance.

2.6.2. True Negative Rate

True Negative Rate (TNR, Selectivity, or Specificity), in our context is, of the calls that did not need an ambulance, what proportion did the model classify correctly?

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

TNR is also not affected by class imbalance because it only considers the classification of the negative (majority) class.

2.6.3. Precision

Precision in our context is, of the ambulances the model would say to send, what proportion were actually needed? Increasing precision would save money.

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}}$$

Note that precision considers elements of both the majority and minority classes, so class imbalance affects the precision, but we can fix that.

2.6.4. Balanced Accuracy and Balanced Precision

Balanced accuracy solves the first problem with accuracy, that it is biased towards the majority class. To balance accuracy, start with the definition of accuracy and scale the majority (negative) class elements (True Negative and False Positive) by the proportions of the positive to negative class, P/N. The results turns out to be the average of the true positive rate and the true negative rate (derivation in the technical paper). That TPR and TNR are unaffected by class imbalance confirms that balanced accuracy is unaffected.

$$\text{Balanced Accuracy} = \frac{\left(\text{TN} \cdot \frac{\text{P}}{\text{N}}\right) + \text{TP}}{\left(\text{TN} \cdot \frac{\text{P}}{\text{N}}\right) + \left(\text{FP} \cdot \frac{\text{P}}{\text{N}}\right) + \text{FN} + \text{TP}} = \frac{\text{TPR} + \text{TNR}}{2}$$

In the same way, we can make a balanced precision. It can also be written in terms of TPR and TNR, (derivation in the technical paper), or more naturally in terms of TPR and the false positive rate (FPR), the proportion of the negative (minority) class that the model misclassified as positive. In our application, FNR is the proportion of crash persons who did not need an ambulance to whom the model recommends that we should send an ambulance. We have not seen this metric in the literature. We developed it because balanced accuracy is used extensively in the literature, and it seemed natural to want to balance a useful metric (precision) in the same way that accuracy has been balanced.

$$\text{Balanced Precision} = \frac{\text{TP}}{\text{TP} + \left(\text{FP} \cdot \frac{\text{P}}{\text{N}}\right)} = \frac{\text{TPR} + (1 - \text{TNR})}{\text{TPR}} = \frac{\text{TPR} + \text{FPR}}{\text{TPR}}$$

2.6.5. F1 and G-Mean

These two popular metrics are each the combination of two other metrics.

F1 is the harmonic mean of Precision and Recall. It is often weighted to emphasize one or the other; if $\alpha = 0.5$, the weighted is the same as the original. We can also substitute Balanced Precision (above) to make a Balanced F1.

$$\text{F1} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

$$\alpha\text{-weighted F1} = \frac{1}{\frac{\alpha}{\text{Precision}} + \frac{1-\alpha}{\text{Recall}}}$$

$$\text{Balanced F1} = \frac{2}{\frac{1}{\text{Balanced Precision}} + \frac{1}{\text{Recall}}}$$

Gmean is the geometric mean of Precision and Specificity (TNR).

$$\text{Gmean} = \sqrt{\text{Precision} \times \text{Specificity}} = \sqrt{\frac{TP}{TP + FP} \times \frac{TN}{TN + FP}}$$

In this paper we have a binary classification model, but Liu, Lyu, Liu and Cao (2021) adapted class-weighted F1 for classification of multiple classes in a multi-modal recommendation system. That paper has an excellent discussion of why and how they applied the class-weighted F1 metric in post-processing, because the metric is not differentiable.

2.6.6. Over-Predicting the Minority Class [Needs Citations]

One tests a model by using it to classify the samples in a test subset of the data. For each sample, the model gives a probability $Pr \in (0, 1)$ that the sample belongs to the positive class. Ideally, most of the samples will fall close to the extremes, clearly classified as either positive or negative, indicating that our model is strong and is more likely to correctly classify future samples, but in practice many samples will not be so clearly classified.

The bar graph below illustrates one model classifying the samples of the CRSS dataset into negative (red, ambulance not needed, majority class) and positive (blue, ambulance needed, minority class). The horizontal axes represents the probability $Pr \in (0, 1)$ given by the model that a sample belongs to the class, and the vertical is the number of samples that fall in each probability range.
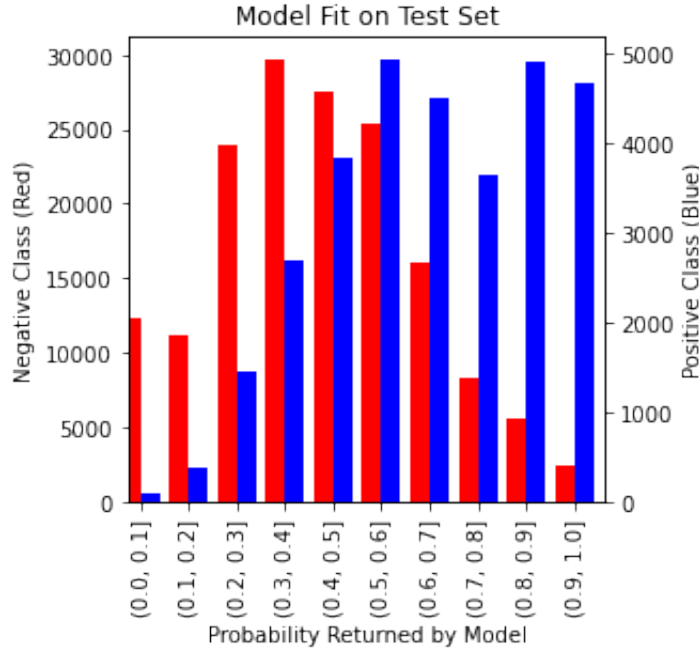
By default, one uses a threshold $Pr = 0.5$ to assign the samples to the negative (majority, $Pr < 0.5$) or positive (minority, $Pr > 0.5$) classes, but we are free to change that threshold. In the bar graph, all red elements to the left of the threshold are TN, all red elements to the right are FP, all blue elements to the left are FN, and all blue elements to the right are TP. The threshold is a hyperparameter that affects all of the above metrics; for example, decreasing the threshold would shift samples from TN to FP, and from FN to TP. Decreasing the threshold would over-predict the minority class, and increasing it would under-predict.

The chart below gives metrics on that model on the CRSS dataset for different thresholds t. The last column is the marginal tradeoff between unneeded ambulances sent and needed ambulances not sent; for instance, at threshold $p = 0.50$, the difference between $p = 0.45$ and $p = 0.55$ is 5.99 ambulances unnecessarily sent for every needed ambulance not sent. If we (as a society) believe that that cost is too high and we should send fewer ambulances, we should choose a larger threshold value and under-predict the minority class. For the purposes of this paper we have arbitrarily chosen 10 as the number of additional false alarms we are willing to tolerate for each additional ambulance sent when needed, so we could change the threshold to somewhere in $0.3 < t < 0.4$, over-predicting the minority class.

t	TN	FP	FN	TP	TPR	FPR	$\dfrac{\Delta\text{TPR}}{\Delta\text{FPR}}$	$\dfrac{\Delta\text{FP}}{\Delta\text{TP}}$
0.00	0	162200	0	31083	1.00	1.00	0.02	246.76
0.10	12322	149878	84	30999	1.00	0.92	0.09	58.08
0.20	23556	138644	455	30628	0.99	0.85	0.26	20.33
0.30	47541	114659	1895	29188	0.94	0.71	0.37	14.10
0.40	77254	84946	4590	26493	0.85	0.52	0.60	8.74
0.50	104764	57436	8425	22658	0.73	0.35	0.87	5.99
0.60	130098	32102	13362	17721	0.57	0.20	1.23	4.25
0.70	146094	16106	17860	13223	0.43	0.10	1.77	2.95
0.80	154351	7849	21506	9577	0.31	0.05	3.09	1.69
0.90	159855	2345	26404	4679	0.15	0.01	8.00	0.65
1.00	162200	0	31083	0	0.00	0.00	14.18	0.37

### 2.6.7. Area Under the ROC Curve [Needs Citations]

The Receiver Operating Characteristic (ROC) shows how much the positive and negative classes are interleaved. It is a parameterized curve following the probability threshold from $t = 0$ to $t = 1$, and plotting the true positive rate

(TPR) versus the false positive rate (FPR). The orange "Model" curve in the graph below is from the same dataset predictions in the chart above. The eleven points correspond to the (FPR,TPR) coordinates in the chart, going from $t = 0.0$ in the upper right to $t = 1.0$ in the lower left. The next-to-last column in the chart gives the slope of the ROC curve.

If the model gave entirely random results, the bars in the bar chart above would all have the same height (1/20 of the number of elements in the class), and the slope of the ROC curve would be 1, illustrated by the green line, and the area under the curve (AUC) would be 0.5. If our model and dataset were ideal, with all of the negative elements having $Pr = 0.0$ and all of the positive elements having $Pr = 1.0$, the ROC curve would be the blue Γ, with area under the curve of 1.0. In practice, the area under the curve is between 0.5 and 1.0.

The area under the ROC curve is useful for comparing different models on the same dataset. A higher area generally means that the model more clearly separates the two classes, and is more likely to correctly classify elements of a different (but related) set of samples.

Because both TPR and FPR are unaffected by class imbalance, the ROC curve is as well.
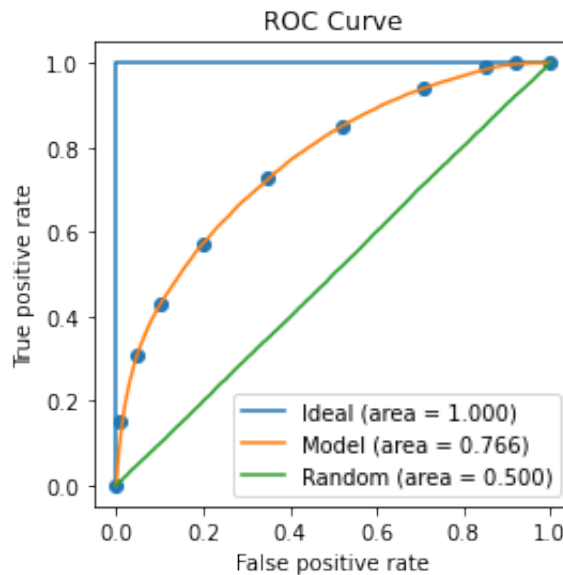
2.7. Imbalanced Data in Crash Analysis

Imbalanced data is a frequent concern in crash analysis. In crash prediction, "non-crash" samples are much more numerous than "crash" samples.

2.7.1. Oversampling

Parsa, Taghipour, Derrible and Mohammadian (2019) used SMOTE to balance the dataset of crashes on Chicago's Eisenhower expressway. Li, Abdel-Aty and Yuan (2020) used SMOTE to study crashes on urban arterials. Guo, Zhao, Yao, Yan, Su, Bi and Wu (2021) used SMOTE to consider risky driving behavior in crash prediction. Orsini, Gecchele, Rossi and Gastaldi (2021) used SMOTE in building a a real-time conflict prediction model. Elamrani Abou Elassad, Mousannif and Al Moatassime (2020) used SMOTE in studying crash prediction for collision avoidance systems. Morris and Yang (2021) compared three oversampling methods, random over-sampling, SMOTE, and adaptive synthetic sampling, for crash data analysis. Yahaya, Guo, Fan, Bashir, Fan, Xu and Jiang (2021a) compared three oversampling methods studying contributing factors in fatal crashes in Ghana, and the same group applied the majority weighted minority oversampling (MWMOTE) method to study multiple fatal injury crashes (Yahaya, Guo, Jiang, Bashir, Matara and Xu, 2021b).

2.7.2. Multiple Techniques

Schlögl, Stütz, Laaha and Melcher (2019) used SMOTE and maximum dissimilarity undersampling to balance an Australian dataset to build a model to predict crashes, while in a later paper Schögl used a balanced bagging approach (Schlögl, 2020). Chen, Shi, Wong and Yu (2020) used SMOTE and Tomek's links in studying crash

potential in lane-changing behavior. Peng, Li, Wang, Gao and Yu (2020) used undersampling, SMOTE, cost-sensitive algorithms, and boosting. Li, Li, Yuan, Lu and Abdel-Aty (2021) used the proximity weighted synthetic oversampling technique (ProWSyn) method to build a traffic violation prediction model. Chen, Lu, Fu, Sze and Ding (2022a) compared Synthetic Minority Over-Sampling Technique for panel data (SMOTE-P) with Random Under-sampling of the Majority Class (RUMC) technique, Cluster-Based Under-Sampling (CBUS), and mixed resampling to identify explanatory factors that affect the crash risk of buses in Hong Kong. Chen et al. (2020) used ENN-SMOTE-Tomek Link (EST) in estimating crash risk in lane changing.

2.7.3. Image (or Image-like) Data

Formosa, Quddus, Ison, Abdel-Aty and Yuan (2020) used Deep Neural Network (DNN) models with a Mean Squared False Error loss function to analyze images from front-facing cameras in cars on a UK roadway to predict crashes. Lin, Li, Jing, Ran and Sun (2020) used Generative Adversarial Networks (GAN) to overcome data imbalance for their incident detection model. Islam, Abdel-Aty, Cai and Yuan (2021) found that a variational autoencoder was more useful than SMOTE, ADASYN, and GAN for generating minority samples to balance crash and non-crash events. Basso, Pezoa, Varas and Villalobos (2021) used Deep Convolutional Generative Adversarial Networks technique with random undersampling to use image and image-like data to build an accident-prediction model for a section of highway in Chile. Man, Quddus and Theofilatos (2022) used Wasserstein GAN (WGAN) and random undersampling to study the transferability of a model built on one dataset to other datasets. Cai, Abdel-Aty, Yuan, Lee and Wu (2020) compared deep convolutional generative adversarial network (DCGAN) with SMOTE and random undersampling to study the effects of proactive traffic safety management strategies such as variable speed limits and dynamic message signs.

2.7.4. Other

Lack et al. (2021) used bagging in predicting crashes for trucks and finding ways to improve truck safety. Haule, Ali, Alluri and Sando (2021) used boosting in studying the effects of ramp metering on traffic safety. Yu et al. (2020) implemented the new focal loss technique in real-time crash prediction. Shi, Wong, Li, Palanisamy and Chai (2019) used undersampling to analyze factors predicting risky and safe driving. Zhu and Wan (2021) used cost-sensitive semi-supervised logistic regression (CS3LR) for hit-and-run analysis.

2.7.5. Imbalanced Data in Other Transportation Areas

Mohammadi, He, Ghofrani, Pathak and Aref (2019) used the Adaptive Synthetic Sampling Approach (ADASYN) on a dataset of foot-by-foot track geometry and tonnage to identify the factors that predict rail defects. Shi and Zhang (2021) developed a hierarchical over-sampling bagging method based on Grey Wolf Optimizer (GWO) algorithm and Synthetic Minority Over-sampling Technique (SMOTE) to study lane changing for autonomous vehicles. The data was severely imbalanced because lane changing is rare compared with lane keeping. Khan, Ma, Chung and Wen (2021) used SMOTE and Tomek links and "average balanced recall accuracies" for flight delay prediction. Chen, Liu, Wang and Yamamoto (2022b) used bagging for ride-hailing demand prediction.

In crash severity, most reported crashes are just property damage only (PDO), and many PDO crashes aren't even reported. Other aspects of transportation also have imbalanced data, including

- Jiang, Yuen and Lee (2020) used similar data and addressed the challenges we'll have with it.

- Elamrani Abou Elassad et al. (2020) works with several imbalanced methods. Use this paper as a model.

2.7.6. Ambulances
- Park and Oh (2019) has a full-page table categorizing studies of ambulance location, relocation, and dispatching using different optimization methods.

3. Methods

We used the Crash Report Sampling System (CRSS) NHTSA (2016-2020) data to train and test our model. After preparing the data, we built a variety of models with different combinations of techniques for handling imbalanced data, model types, loss functions, and sets of features, with the overall goal of finding an optimal combination of methods to give the most useful model.

3.1. Dataset Overview

The Crash Report Sampling System (CRSS) (NHTSA, 2016-2020) is an annually published anonymized sample of crash reports in the US, produced by the National Center for Statistics and Analysis (NCSA) of the National Highway Transportation Safety Administration (NHTSA) of the US Department of Transportation (DOT). CRSS replaced the National Automotive Sampling System (NASS) General Estimates System (GES) in 2016. This paper uses 2016-2020 data; the 2021 data is due out in spring 2023.

For each year, the CRSS data comes in three main files, the first with a row for each crash (`Accident.csv`), the second with a row for each vehicle involved in the crash (`Vehicle.csv`), and the third with a row for each person involved in the crash (`Person.csv`). Each year also has (depending on the year) about twenty other files with more detail, which we did not investigate. Putting the five years together, the `Accident` file has 51 features with 259,077 samples, the `Vehicle` has 97 features with 457,314 samples, and the `Person` has 67 features with 644,274 samples. Some features were added or discontinued during these five years.

All of the data is discrete or categorical. Some of the features have ordered data (year, month, hour), but some of them are unordered (make of vehicle). Most features have missing data with a value (like 9 or 99) to signify "missing." Some features had their missing values imputed (in a separate feature, so we know which samples had the value missing).

Accompanying the CRSS data are a very useful Analytical Users Manual (National Center for Statistics and Analysis, 2022) and a report on how (and why) they imputed missing values in some but not all features. (Herbert, 2019)

3.2. Preparing the Data

To prepare the data for building a model we have two challenges. First, the data has missing values, and we need to choose an approach to handling them. Second, some features have too many unique values for effective model building, and we need to group (bin, discretize) them into larger categories.

To handle the missing values, we have three options, to throw out the samples with missing values in the features we are using to build the model (around 30% of the samples), to treat "missing" as its own value in each feature, or to use some statistical or machine learning method to impute missing values. We chose to impute missing values.

To impute missing values on some features, the CRSS authors used IVEware: Imputation and Variance Software, which uses multivariate sequential regression, and the CRSS imputation report gives the values for the hyperparameters they used. (Herbert, 2019) Of the forty-three features we want to use for our most robust model, only eight were imputed in CRSS; the other thirty-five features have one or more values signifying "Missing."

Why did the CRSS authors not impute missing values in all of the features? For some features it is the nature of the data. In other fields, like VIN (Vehicle Identification Number), if that data is missing from the report, there is no way to recover, or even guess at, that information using other data in the sample. On the bureaucratic form that is a police crash report, the police jurisdiction is usually printed on the form, and the date is one of the first fields that the reporting officer understands is vital. No CRSS samples have the month or day of week missing; however, CRSS does impute `DAY_WEEK` as `WKDY_IM`, with exactly the same values. The fuller logic for which fields get imputed is partly the nature of the data but also backwards historical compatibility with CRSS's predecessor databases going back to 1982. See the CRSS technical report on imputation for fuller details. (Herbert, 2019)

An interesting question arises, of whether the order of operations matters. Does it make a difference whether we impute first, then bin, or bin first, then impute? We tested by removing all samples with a missing value, deleting 15% of known values, running impute/bin and bin/impute, then comparing the success rates of the imputations. We found that the difference between the two methods is negligible, that the randomness caused as much difference as the order of operations. For full details see the attached code, `CRSS_11_09_22_Bin_Impute.ipynb`. We decided to bin then impute, because IVEware could not handle categorical features with more than about forty unique values, so we had to bin some of the features before imputing anyway.

We compared three methods for imputing missing values. The first method was to impute to mode: Assign to each missing value the most common value in that feature. This method is not only easy but widely used. The second method was IVEware, using the same hyperparameters as the CRSS authors used. The third method was Random Forest Classifier, using the mode as the starting guess. Random Forest and IVEware were both better than Mode, and Random Forest was arguably better than IVEware, but not by much and not in all features. Both Random Forest and IVEware correctly imputed about 77% of missing values. For full details, see the attached code, `CRSS_05_Impute_Random_Forest_12_22_22`.

To bin the values in each feature into fewer categories, for some features the best binning was straightforward. The HOSPITAL feature in the Person dataset, which we made the target variable for our models, has nine values, one signifying that the person did not go to the hospital, six telling how the person got to the hospital (by air, by law enforcement, by emergency medical services but unknown mode, by EMS ground, transported by other method, or other), and two values signifying that the report didn't say whether the person went to the hospital. For our purposes, it does not matter how the person got to the hospital, only that the person went. We reduced the nine values to three: Hospital True, Hospital False, and Hospital Unknown. Many features were this straightforward.

Other features were like MAKE, which has sixty-seven unique values with no clear order. The chart below shows the major maufacturers. To avoid the curse of dimensionality when we change the bins to dummy variables, we ideally want to put the sixty-seven values into five to ten bins, but does Toyota go in the same bin with Ford or Hyundai? We imposed an ordering on the set, based on how likely someone in that make of car was to go to the hospital. We then put them into nine bins, and one bin for "Unknown Make." we divided the nine bins first along significant changes in the percent hospitalized, like the one between the motorcycles and the passenger cars and the one between the cars and the heavy trucks, and then into approximately even numbers of persons in each group when there was no clear line.

Note that "Bluebird" here is a school bus, not the Nissan Bluebird. Also, some of the makes, like Volvo and Mercedes-Benz, include both passenger cars and heavy trucks.

The feature MAK_MOD, make and model, has 1200 categories that we similarly mapped into five bins of approximately equal number of persons by their hospitalization rates.

For full details on the binning, see the code, CRSS_04_Discretize_11_19_22.

MAKE	Percent of Persons	Percent of those Persons Hospitalized
Yamaha	0.33	63.16
Harley-Davidson	0.82	61.86
Kawasaki	0.26	60.61
Suzuki	0.44	45.11
Other Make	0.63	30.46
Mitsubishi	0.78	16.66
Pontiac	0.87	16.56
Mercury	0.54	16.29
Buick / Opel	1.36	16.07
Saturn	0.50	15.89
Honda	8.85	15.81
KIA	2.95	15.46
Nissan/Datsun	7.71	15.23
Chrysler	1.95	15.07
Fiat	0.06	15.00
Hyundai	3.67	14.87
Chevrolet	11.93	14.25
Lincoln	0.56	13.99
BMW	1.21	13.69
Toyota	11.09	13.34
Volkswagen	1.42	13.29
Ford	12.67	13.16
Cadillac	0.87	12.95
Jeep	3.03	12.76
Mazda	1.4	12.70
Infiniti	0.82	12.66
Dodge	5.56	12.44
Mercedes-Benz	1.23	12.39
Acura	0.94	12.35
GMC	2.49	11.95
Lexus	1.34	11.58
Subaru	1.22	10.58
Audi	0.48	9.59
Volvo	0.57	9.02
Bluebird	0.05	7.88
Thomas Built	0.02	7.52
Mack	0.17	4.35
International Harvester/Navistar	0.48	4.06
Freightliner	0.85	4.03
Kenworth	0.26	3.92
Peterbilt	0.31	3.11
Not Reported	0.27	1.20
Unknown Make	2.07	3.24

3.3. Feature Selection

In selecting features from the CRSS dataset, we chose features that could be known or inferred at the time of the crash without an eyewitness. We separated them into three sets in terms of how hard the data would be to get in good quality: Easy, medium, and hard. The easy information is a baseline, the information the police have before they receive the notification, like locality, time of day, and weather. The medium level requires detailed maps to distinguish

between, for example, an intersection and a parking lot, and information from the phone service provider about the primary user of the phone. The hard level would require coordination of private (cell service providers, Apple Google) and public databases (police, government vehicle registries), or correlating several notifications from different phones at about the same time from about the same location to see that they are probably the same crash event.

Location

Easy

Region (4 regions of the US)

Principal Sampling Unit (74 subregions of the US)

Police Jurisdiction (422 jurisdictions)

Urban/Rural

Medium: Requires precise location and maps

Interstate

In intersection (Yes/No)

Type of interesection

Position relative to the road

Position relative to an intersection

Roadway alignment (curve, straight, driveway,...)

Number of lanes in roadway

Roadway grade

Speed limit

Traffic control device (stop lights, stop signs)

Trafficway description (two way, divided, ramp,...)

Medium: Requires up-to-date maps

Work Zone

Hard: Requires very detailed maps

Relation to junction features

Time

Easy

Hour

Weekday/Weekend

Month

Year

Weather

Medium: Requires detailed maps per time of day

Lighting Conditions

Phone User

Medium: Requires identifying phone user

Age

Sex

Medium: Requires storing acceleration data for several minutes

Pedestrian / In moving vehicle

Hard: Requires identifying the vehicle used by phone user

Make of vehicle

Model of vehicle

Age of vehicle

Body type of vehicle

Hard: Requires identify licensing of phone user (driver) or ages from several phones (bus passengers)

School bus

Type of other bus

Emergency vehicle

Correlate Multiple Notifications

Hard

Number of people in vehicles

Number of occupants of particular vehicle

Number of vehicles in motion

Number of parked vehicles

3.4. Feature Engineering

From the features in the dataset, we can create new features that may be relevant, like "Rush Hour," combining hour with day of week. During rush hour, the number of crashes increases, but their average severity goes down. We drew the lines distinguishing "Rush Hour" from "Not Rush Hour" based, again, on the correlation to hospitalization. Interestingly, the hospitalization rates varied by age, and varied by sex, but varied differently by age within sex. Hospitalization rates for young children are low, perhaps due to child seat requirements, but after young childhood, as age increases, the hospitalization rate increases for both men and women, but younger women have a lower hospitalization rate than men, and older women have a higher rate. We created an AGE_SEX feature.

Crash Person Count

Age	Female	Male	Total
00-06	12331	12639	24970
07-15	16623	16726	33349
16-18	20158	21622	41780
19-52	175528	219931	395459
53-70	49750	65228	114978
71+	15972	17766	33738
Total	290362	353912	644274

Hospitalization Count

Age	Female	Male	Total
00-06	1448	1560	3008
07-15	2529	2779	5308
16-18	2900	2998	5898
19-52	28843	33514	62357
53-70	9302	10947	20249
71+	3503	3287	6790
Total	48525	55085	103610

Ambulance Dispatch

Hospitalization Rate

Age	Female	Male	Total
00-06	0.117	0.123	0.240
07-15	0.152	0.166	0.318
16-18	0.144	0.139	0.283
19-52	0.164	0.152	0.316
53-70	0.187	0.168	0.355
71+	0.219	0.185	0.404
Total	0.983	0.933	1.916

Rate Normalized by Row

Age	Female	Male	Total
00-06	0.488	0.512	1.0
07-15	0.478	0.522	1.0
16-18	0.509	0.491	1.0
19-52	0.519	0.481	1.0
53-70	0.527	0.473	1.0
71+	0.542	0.458	1.0
Total	3.063	2.937	6.0
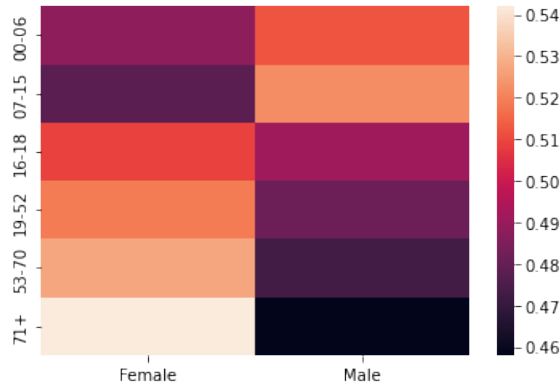
Hospitalization Rate Normalized by Age

For full details see the code `CRSS_05_24_22_Crosstabs`.

3.5. Handling Data Imbalance and Building Models

In the Literature Review section we enumerated many methods for dealing with data imbalance, most of which are relevant to our problem. In the code `CRSS_06_Build_Model_01_15_22.ipynb` we implemented these methods. We tried all of them, individually and in combinations, to see which combinations of methods give the most effective model.

- Ensure that the training and test sets have the same proportion of positive and negative class samples

- Tomek Links, both once and twice

- Balanced Metrics

  - Balanced Accuracy
  - Balanced F1
  - Balanced Precision

- Loss Functions

    - α-weighted Binary Crossentropy

    - Focal Loss

- Model Algorithms

    - Keras Simple Neural Network

    - AdaBoost Classifier

    - Balanced Bagging Classifier

    - Balanced Random Forest Classifier

3.6. Class Weights as Ethical Tradeoff Rate

We can use the class weight hyperparameter to incorporate into the model our value judgement about, at the margin, how many ambulances we are willing to send on a wild goose chase in order to send one that is needed, which is a judgement about how many dollars a life is worth.

Many loss functions incorporate a class weight hyperparameter, here given by α. One of its uses is to accommodate class imbalance, described above, where we let $1/r$ be the proportion of samples in the minority class.

$$\text{Let } r = \frac{\text{Total number of samples}}{\text{Number of minority samples}} \qquad \text{Let } \alpha = \frac{r}{r+1} \qquad 1 - \alpha = \frac{1}{r+1}$$

The loss function for each sample is given by L, and the total loss is the sum of those sample losses, J.

$$L(y, p, \alpha) = -\left(\alpha y \log(p) + (1 - \alpha)(1 - y)\log(1 - p)\right)$$

$$J(y, p, \alpha) = -\sum_{i=1}^{N} \left(\alpha y_i \log(p_i) + (1 - \alpha)(1 - y_i)\log(1 - p_i)\right)$$

Let us recall the confusion matrix, in terms of y_i and p_i. We will use it to switch between binary and continuous versions of the loss functions.

	Do Not Send Ambulance $p_i \leq 0.5$	Send Ambulance $p_i > 0.5$
Ambulance Not Needed $y_i = 0$	TN	FP
Ambulance Needed $y_i = 1$	FN	TP

In the (unweighted) binary cross-entropy loss function,

$$J = -\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

the y_i are binary, $y_i \in \{0, 1\}$, but the model predictions, p_i, are a probability, $p_i \in (0, 1)$.

If we treat the model predictions as binary, replacing

$$\log(p_i) \rightarrow \begin{cases} 0 & \text{if } p_i \leq 0.5 \\ 1 & \text{if } p_i > 0.5 \end{cases} \qquad \text{and} \qquad \log(1 - p_i) \rightarrow \begin{cases} 0 & \text{if } 1 - p_i \leq 0.5 \\ 1 & \text{if } 1 - p_i > 0.5 \end{cases}$$

then

$$TP = \sum_{i=1}^{N} y_i \log(p_i) \qquad \text{and} \qquad TN = \sum_{i=1}^{N} (1 - y_i)\log(1 - p_i)$$

and the loss function becomes $J = -(TP + TN)$.

We use the continuous version when building the model, because we want the predictions to be robust, so that when we use the model on unseen data we can be more certain that it will correctly classify new instances. We use the binary when we evaluate the model on unseen data, because then we only care about whether the model gets the classification right or wrong. The binary is also easier to explain.

If the medical ethicists and politicians decide on a tradeoff threshold, r such that, at the margin, we are willing to automatically dispatch r ambulances when they aren't needed in order to send one ambulance when it is needed, then we want

$$\frac{\Delta FP}{\Delta TP} \leq r$$

which makes the binary version of our loss function $FP - r \cdot TP$, and the continuous version equivalent to the α-weighted cross-entropy loss function.

$$J = -\sum_{i=1}^{N} \alpha y_i \log(p_i) + (1 - \alpha)(1 - y_i) \log(1 - p_i), \quad \alpha = \frac{r}{r + 1}$$

Why are these equivalent?

Adding a constant to the loss function, or multiplying it by a positive constant, does not change its effect, because in comparing one iteration of the model to another, the algorithm is only concerned with which has the smaller loss.

The binary loss function $FP - r \cdot TP$ is equivalent to $FP - r \cdot TP - (TN + FP)$, because $TN + FP$ is the number of negative samples in the dataset (thus constant), so as a loss function, $FP - r \cdot TP$ is equivalent to $-(r \cdot TP + TP)$.

$$FP - r \cdot TP$$

$$-(r \cdot TP + TN)$$

Multiplying by $\frac{1}{r+1}$ gives an equivalent loss function, because $\frac{1}{r+1} > 0$.

$$-\frac{r \cdot TP + TN}{r + 1}$$

$$-\left(\frac{r}{r + 1}TP + \frac{1}{r + 1}TN\right)$$

$$-\left(\frac{r}{r + 1}TP + \left(1 - \frac{r}{r + 1}\right)TN\right)$$

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

The continuous versions of TP and TN are $\sum_{i=1}^{N} y_i \log(p_i)$ and $\sum_{i=1}^{N} (1 - y_i) \log(1 - p_i)$, so we get the α-weighted binary cross-entropy loss function,

$$J = -\sum_{i=1}^{N} \alpha y_i \log(p_i) + (1 - \alpha)(1 - y_i) \log(1 - p_i), \quad \alpha = \frac{r}{r + 1}$$

In training our models we will consider different values for the ethical tradeoff rate r. A very small rate, like $r = 1$ may be ethically justifiable, since our model is recommending whether to send an ambulance *now*, without waiting for more information from eyewitnesses, and police can reassess when they have more information. We will test $r = 1$, $r = 10$, and the rate of class imbalance in the data, about $r = 5$.

3.7. ROC Slope as Ethical Tradeoff Rate

Similarly to choosing the class weight hyperparameter to reflect our value judgement about how many unneeded ambulances we are willing to send to send one that is needed, we can use the slope of the ROC curve to choose the threshold between negative and positive predictions. The class weight hyperparameter has a marginal effect, influencing the loss on each sample as the model is built, and the slope of the ROC curve has a total effect on the model's results.

As before, if the medical ethicists and politicians decide on a tradeoff threshold, r such that, at the margin, we are willing to automatically dispatch r ambulances when they aren't needed in order to send one ambulance when it is needed, then we want $\Delta FP/\Delta TP r$.

$$\Delta \frac{FP}{TP} = r$$

$$TPR = \frac{TP}{P}, \ FPR = \frac{FP}{N}$$

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

3.8. Model Evaluation: Baselines for Comparison

In the supervised learning method we used here, for each of the $\approx 600,000$ samples (people) in the dataset, we know the answer (the *label* or *ground truth*) to the question, whether the person needed an ambulance, $y = 0$ for "no" and $y = 1$ for "yes." We are trying use historical data to build a model to predict the label for new data (incoming automated crash notifications).

The binary classification models we used return, for each sample, a continuous probability $p \in (0, 1)$ that the sample belongs in the positive class. If a sample has $p = 0.1$, the model is 90% confident that this sample is in the negative class. We then pick a threshold, usually but not necessarily 0.5, and make a binary prediction, that samples with $p > 0.5$ need an ambulance, and those with $p < 0.5$ do not. (What happens if $p = 0.50000000000000000$ would apply to negligibly few samples, so which way it goes does not matter.) The *loss function* used by the model is the sum not of how many binary predictions were incorrect, but how strongly incorrect the continuous predictions were. If the prediction for a sample is $p = 0.3$ and the label is $y = 0$, then the loss for that sample is 0.3, but if its label were $y = 1$, then the loss would be 0.7.

A perfect model would not only predict each sample's label correctly, but would do it with perfect certainty. In the real world, with interesting questions about real data, we will have false positives ($y = 0$ and $p > 0.5$) and false negatives ($y = 1$ and $p < 0.5$), but we hope those are few, and that the predictions are strongly correct, meaning the predictions are close to their labels.

When we get results for our models based on crash data, we need some frame of reference for what is "good" and "bad," so we have created some sets of entirely artificial results using a gamma distribution for ideal results and a uniform distribution for awful results.

The histogram below of the percent of samples with predictions p in each range illustrates the best results we can hope for in the real world. The positive class is small because the data is imbalanced. about 15% of the dataset, as in our CRSS data. There are some false positives and negatives, but the overwhelming majority of the predictions are correct, and most with strong confidence.

The Receiver Operating Characteristic (ROC) is a parameterized curve following the probability threshold from $p = 0$ to $p = 1$, plotting the true positive rate (TPR) versus the false positive rate (FPR). The Area Under the ROC curve (AUC) is often used to compare two models, with AUC of 1 indicating perfect prediction and AUC of 0.5 indicating no discernable pattern.

We have added to the typical ROC curve two labels, one for the positive and one for the negative class, of the median of the probabilities of the samples in that class. T

Probability Distribution

ROC Curve
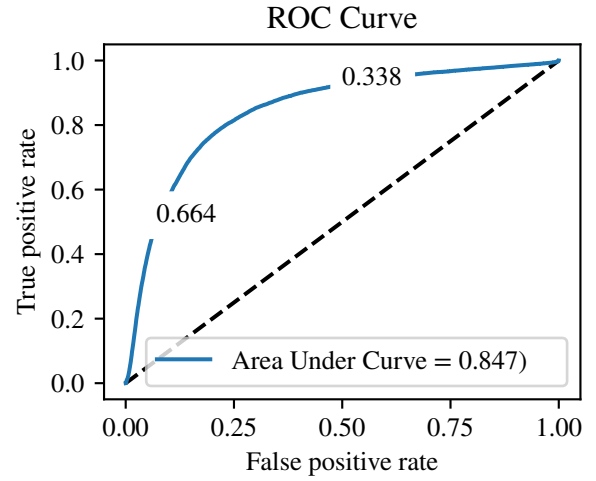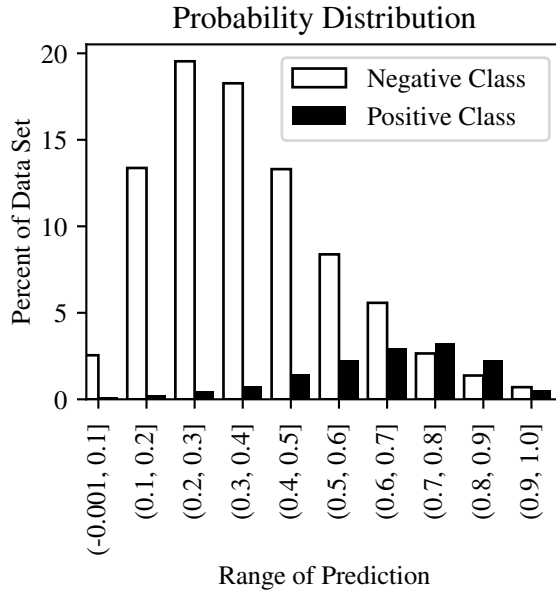
The *confusion matrix* for this ideal data set, here given as percentages of the entire dataset, shows few false positives and false negatives. The metrics below are the ones we will watch when evaluating models. Each of them tells a different story about what the model does well.

		Prediction	
		N	P
Actual	N	67.0%	18.7%
	P	3.03%	11.3%

0.783	Accuracy
0.785	Balanced Accuracy
0.376	Precision
0.783	Balanced Precision
0.788	Recall
0.509	F1
0.785	Balanced F1
0.542	Gmean

If we do not address the data imbalance, the model building algorithm will maximize accuracy by classifying most (or all) of the samples as "No Ambulance" with $p < 0.5$ We built the artificial results below by multiplying the probabilities in the above results by 0.5. Note that the Area Under the Curve (AUC) did not change.

Probability Distribution

ROC Curve
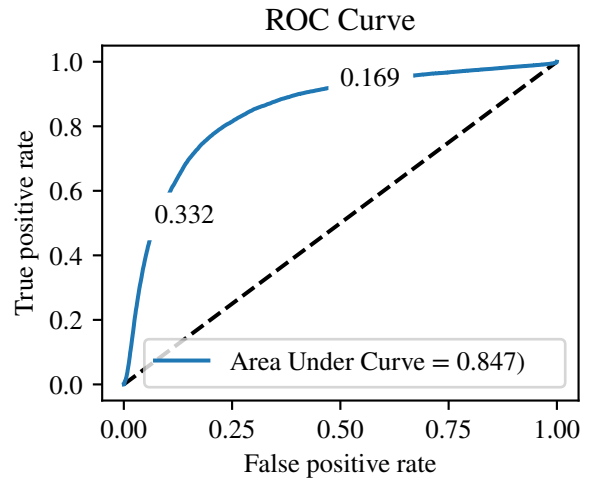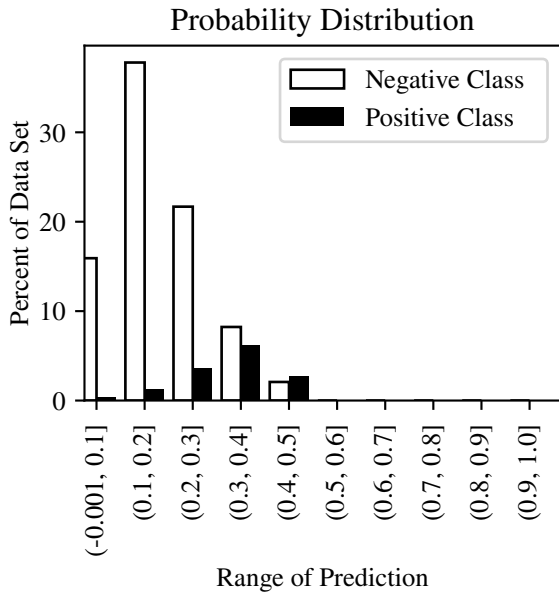
	Prediction	
	N	P

		N	85.7%	0.0%
Actual				
	P	14.3%	0.0%	

0.857	Accuracy
0.500	Balanced Accuracy
0.000	Precision
0.000	Balanced Precision
0.000	Recall
0.000	F1
0.000	Balanced F1
0.000	Gmean

Such a recommendation system ("Never send an ambulance") would be useless, but note that the distribution still separates the negative and positive classes, just not at $p = 0.5$. We can fix that in two ways; the first is to shift the distribution to be centered at $p = 0.5$. By "centered," we mean that the average of the medians of the negative and positive classes (the 0.107 and 0.293 on the ROC curve above) will now be 0.5. Further research can explore whether centering the distribution at the $p = 0.5$ threshold or another value of p is most useful.

Probability Distribution

ROC Curve
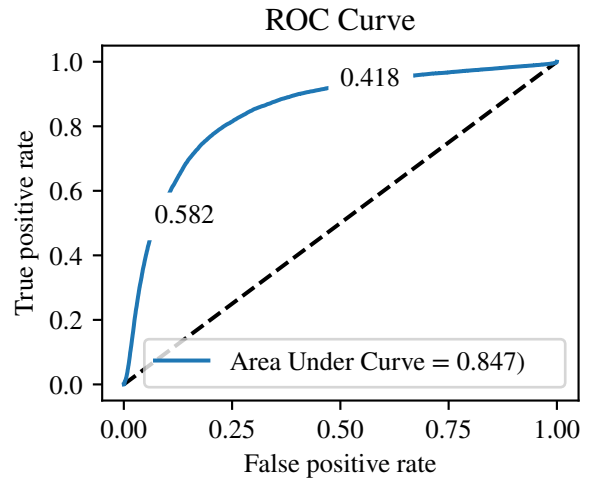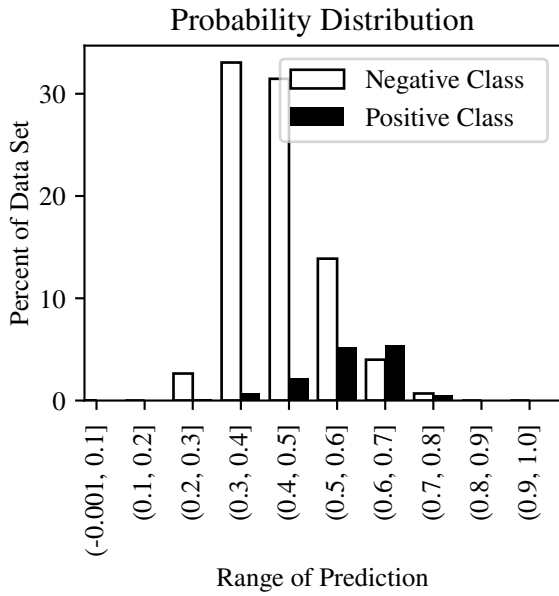
		Prediction	
		N	P
Actual	N	67.2%	18.5%
	P	3.06%	11.22%

0.784	Accuracy
0.785	Balanced Accuracy
0.377	Precision
0.784	Balanced Precision
0.786	Recall
0.510	F1
0.785	Balanced F1
0.543	Gmean

Another way is to linearly transform the probabilities. Whether the distribution was clustered to the left or right, or clustered at the center, is not necessarily relevant, so we want to see it spread out. We have arbitrarily chosen a transformation to put next the original models in our results to see if it will make a better model; tuning the transformation is an avenue for future work. We have chosen to take the 0.05 quantile of the negative class and map it to $p = 0.05$, and the 0.95 quantile of the positive class and map it to $p = 0.95$. This linear transformation gives the same metrics as the shift, and the ROC curve is the same except for the two labeled medians, now at 0.305 and 0.688.

Probability Distribution

ROC Curve
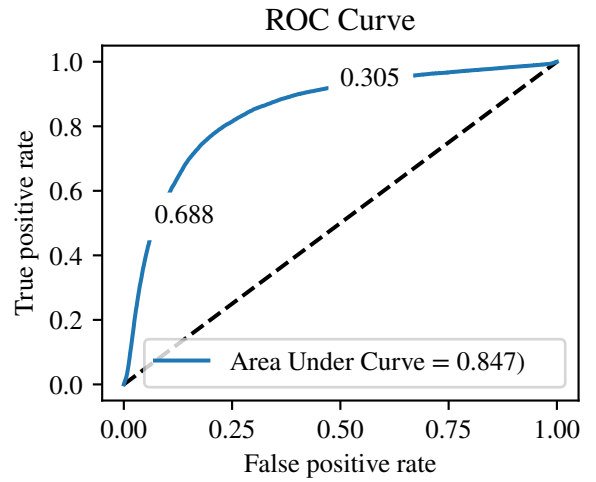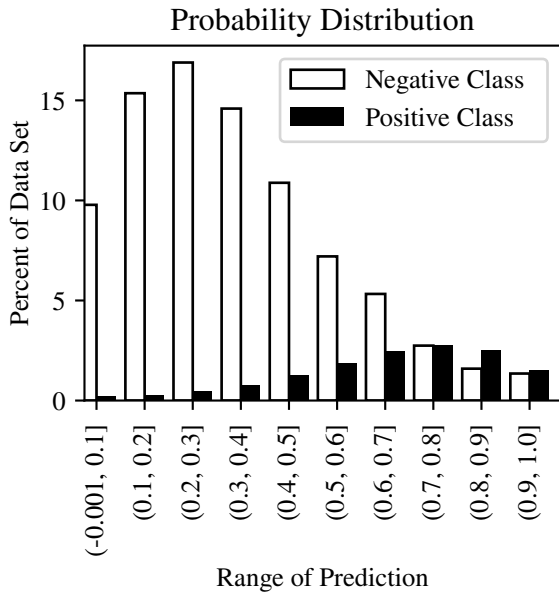
		Prediction	
		N	P
Actual	N	67.5%	18.2%
	P	3.11%	11.2%

0.787	Accuracy
0.785	Balanced Accuracy
0.380	Precision
0.786	Balanced Precision
0.782	Recall
0.512	F1
0.784	Balanced F1
0.547	Gmean

In the ideal results above, the algorithm learned a useful model from the patterns in the data. The results below illustrate the worst case scenario, where the algorithm does not learn a good model, usually because the data does not have a pattern that predicts the target variable. In the ROC curve, the median values of the probabilities for the two classes are so close that the labels are on top of each other.

Probability Distribution

ROC Curve
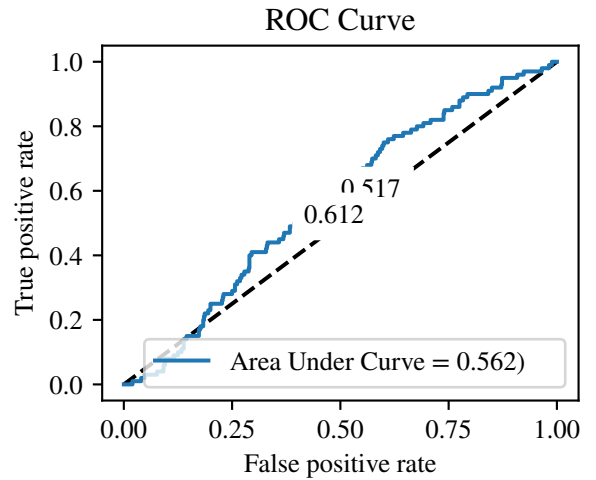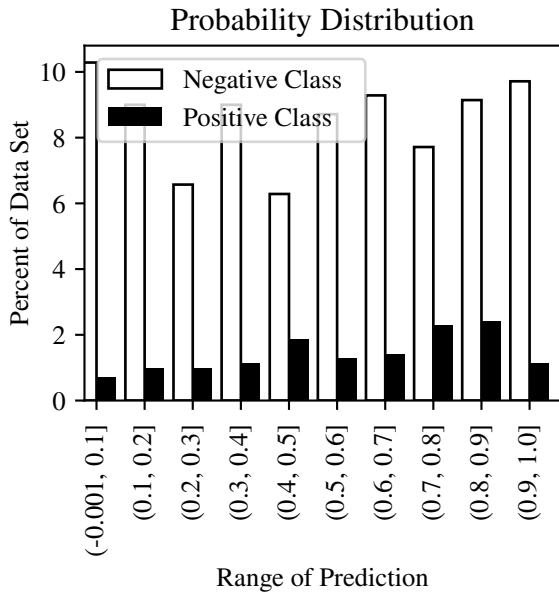
		Prediction	
		N	P
Actual	N	41.1%	44.6%
	P	5.71%	8.57%

0.497	Accuracy
0.540	Balanced Accuracy
0.161	Precision
0.536	Balanced Precision
0.600	Recall
0.254	F1
0.566	Balanced F1
0.278	Gmean

3.9. Model Evaluation: Incorporating Ethical Tradeoffs

In evaluating our models, the considerations are not just technical. A false positive is a recommendation to send an ambulance when one is not needed, which costs money. A false negative is a recommendation to not send an ambulance when one is needed, which costs lives.

Of the metrics we are watching, F1 and AUC seem most useful in discriminating "better" from "worse" models. F1 is the harmonic mean of Precision (What proportion of the ambulances we sent were needed?) and Recall (What proportion of the ambulances needed did we send?), and reflects the discrete results. AUC (Area Under the (ROC) Curve) quantifies the predictive power of the continuous results. It is possible to introduce a parameter to weight the precision and recall parts of F1 if you can quantify how much more important one is than the other, but we had no way to choose that number.

4. Results

We tested several combinations of model inputs.

- Easy, Medium, and Hard to collect features

- Undersampling with Tomek links

- Model types

- Loss Functions

- Class weights parameters and other hyperparameters

We do not hope to get a perfect model, but something better than, "Never send an ambulance," "Always send an ambulance," or random noise.

Each of the five models types we tested had one version in the top five, measured by both F1 and AUC.

F1	AUC	Model
0.429	0.760	Bagging
0.400	0.752	AdaBoost (Linear transformation)
0.364	0.714	Balanced Random Forest
0.354	0.697	α-weighted Binary Crossentropy with Class Weights
0.353	0.695	Binary Focal Crossentropy with Class Weights and $\gamma = 2.0$ (Linear Transformation)

Details follow.

4.1. α-weighted Binary Cross Entropy Model with $\alpha = 0.850, r = 5.66$

Probability Distribution
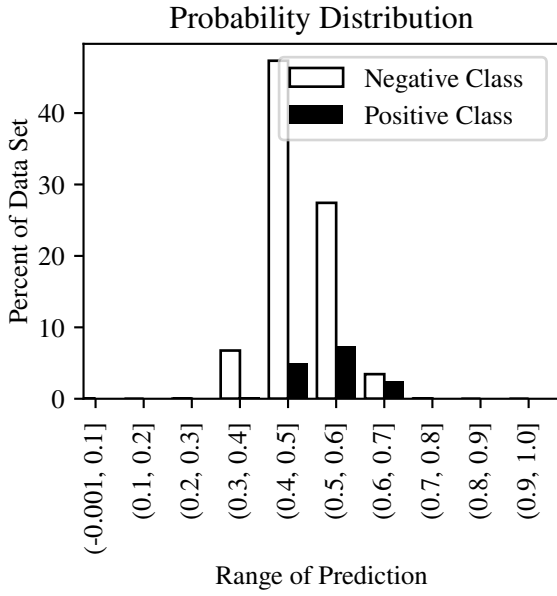
Prediction		N	P
Actual	N	109,166	41,605
	P	11,959	14,662

0.698	Accuracy
0.637	Balanced Accuracy
0.261	Precision
0.666	Balanced Precision
0.551	Recall
0.354	F1
0.603	Balanced F1
0.434	Gmean

Probability Distribution
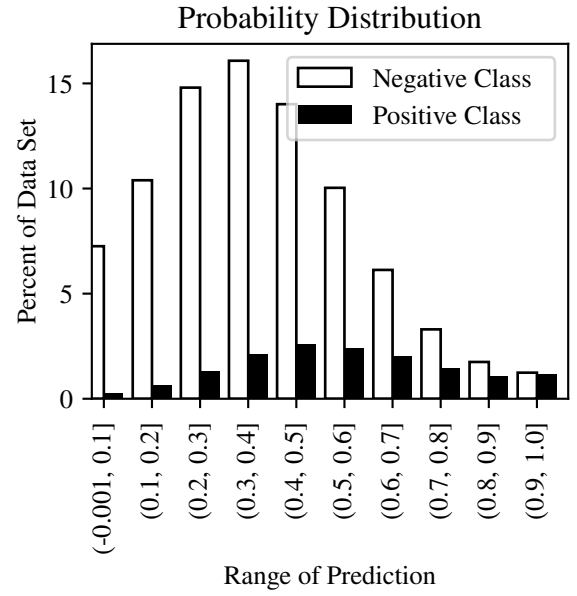
Prediction		N	P
Actual	N	112,677	38,094
	P	12,746	13,875

0.713	Accuracy
0.634	Balanced Accuracy
0.267	Precision
0.674	Balanced Precision
0.521	Recall
0.353	F1
0.588	Balanced F1
0.447	Gmean
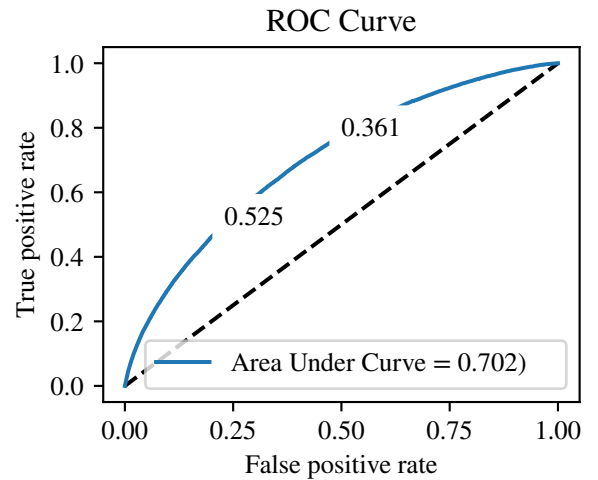
ROC Curve

4.2. α-weighted Binary Cross Entropy Model with $\alpha = 0.5, r = 1$
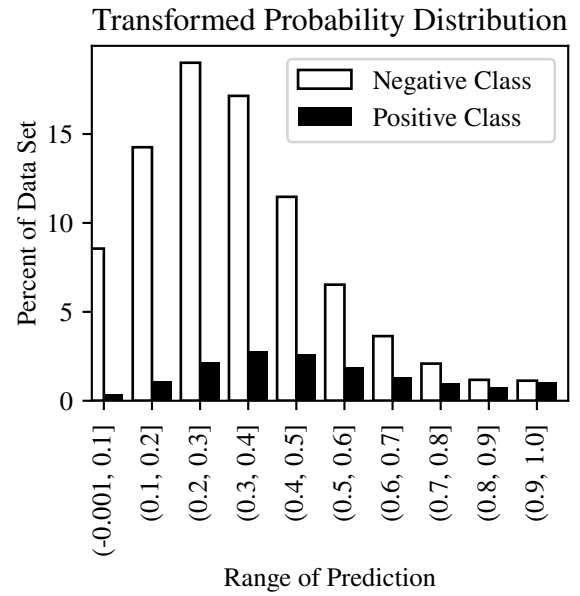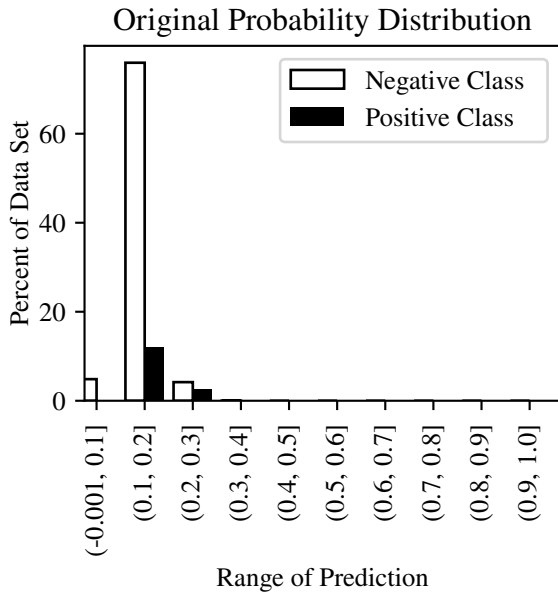
Original Probability Distribution

Transformed Probability Distribution

Prediction		N	P
Actual	N	150,771	0
	P	26,621	0

Prediction		N	P
Actual	N	124,949	25,822
	P	15,894	10,727

0.850	Accuracy
0.500	Balanced Accuracy
0.000	Precision
0.000	Balanced Precision
0.000	Recall
0.000	F1
0.000	Balanced F1
0.000	Gmean

0.765	Accuracy
0.616	Balanced Accuracy
0.293	Precision
0.702	Balanced Precision
0.403	Recall
0.340	F1
0.512	Balanced F1
0.493	Gmean

ROC Curve

4.3. α-weighted Binary Cross Entropy Model with $\alpha = 0.89$, $r = 10$

Original Probability Distribution

Transformed Probability Distribution
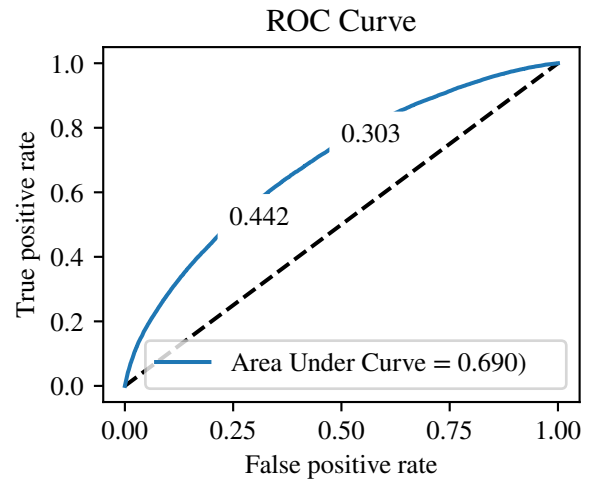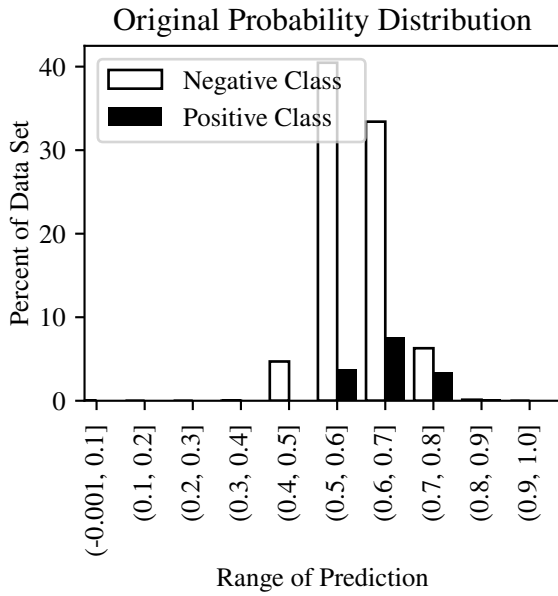
Prediction		N	P
Actual	N	8,346	142,425
	P	210	26,411

0.196	Accuracy
0.524	Balanced Accuracy
0.156	Precision
0.512	Balanced Precision
0.992	Recall
0.270	F1
0.676	Balanced F1
0.093	Gmean
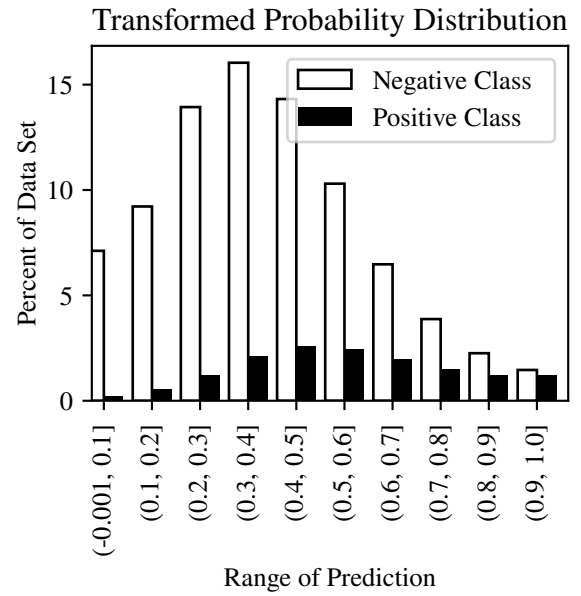
Prediction		N	P
Actual	N	107,543	43,228
	P	11,819	14,802

0.690	Accuracy
0.635	Balanced Accuracy
0.255	Precision
0.660	Balanced Precision
0.556	Recall
0.350	F1
0.603	Balanced F1
0.427	Gmean

ROC Curve
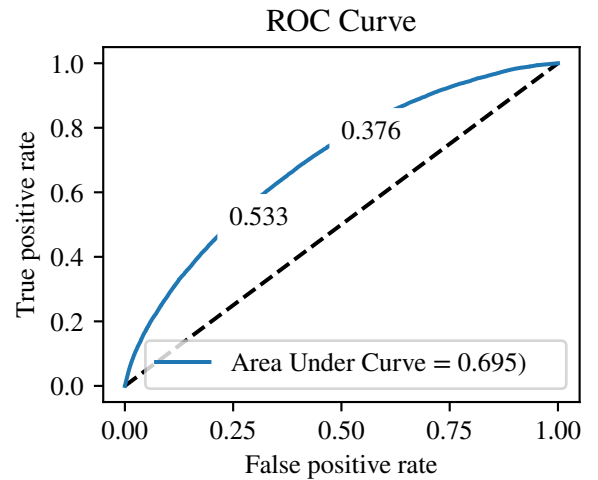
4.4. Binary Focal Crossentropy with $\alpha = 0.850$ and $\gamma = 0.0$
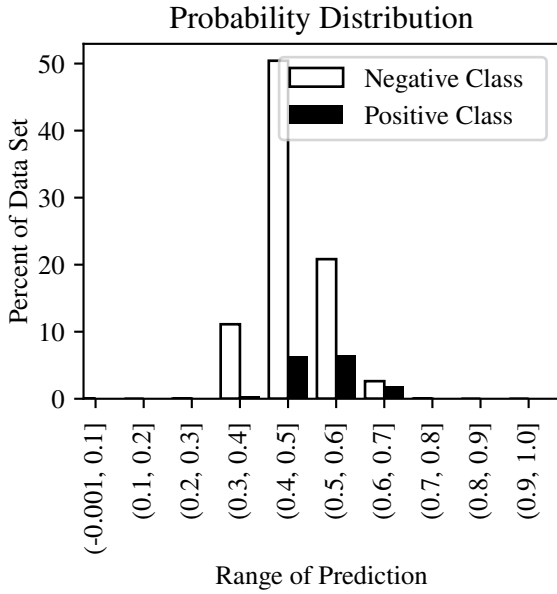
Probability Distribution

Probability Distribution

Prediction		N	P
Actual	N	150,771	0
	P	26,621	0

0.850	Accuracy
0.500	Balanced Accuracy
0.000	Precision
0.000	Balanced Precision
0.000	Recall
0.000	F1
0.000	Balanced F1
0.000	Gmean
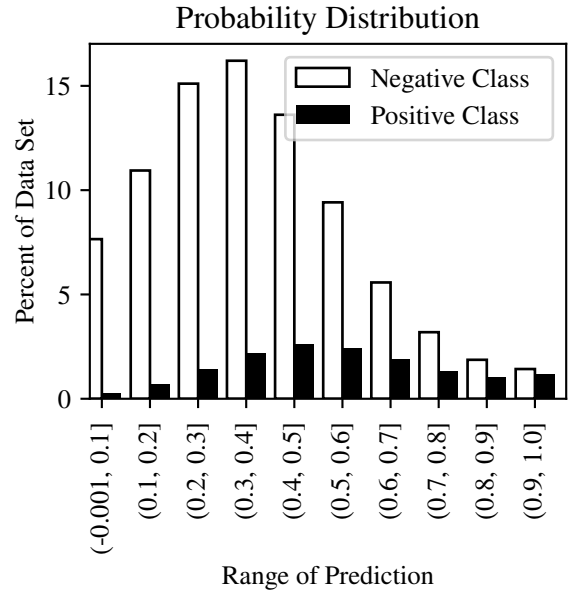
Prediction		N	P
Actual	N	126,475	24,296
	P	16,054	10,567

0.773	Accuracy
0.618	Balanced Accuracy
0.303	Precision
0.711	Balanced Precision
0.397	Recall
0.344	F1
0.510	Balanced F1
0.504	Gmean

The results from this model should be the same as for our original α-weighted binary crossentropy model with $\alpha = 0.850$, but they're not. Need to fix that.
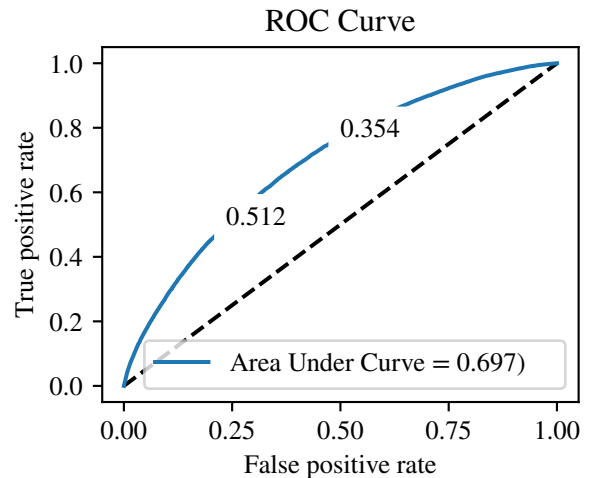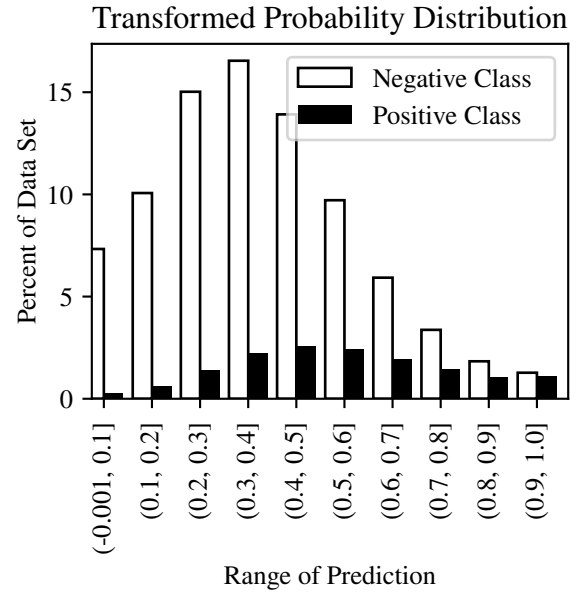
ROC Curve

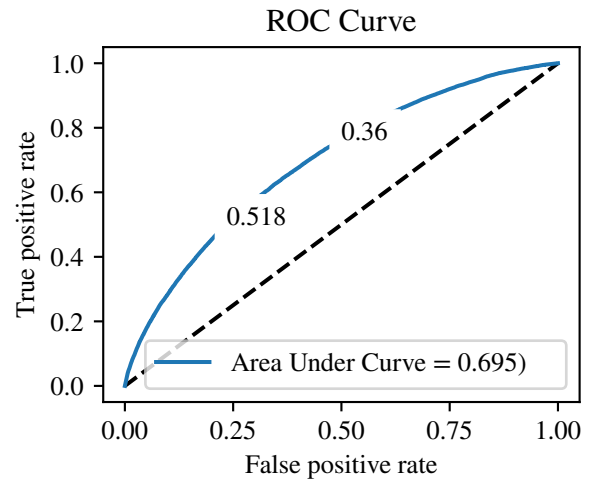## 4.5. Binary Focal Crossentropy with $\alpha = 0.850$ and $\gamma = 2.0$
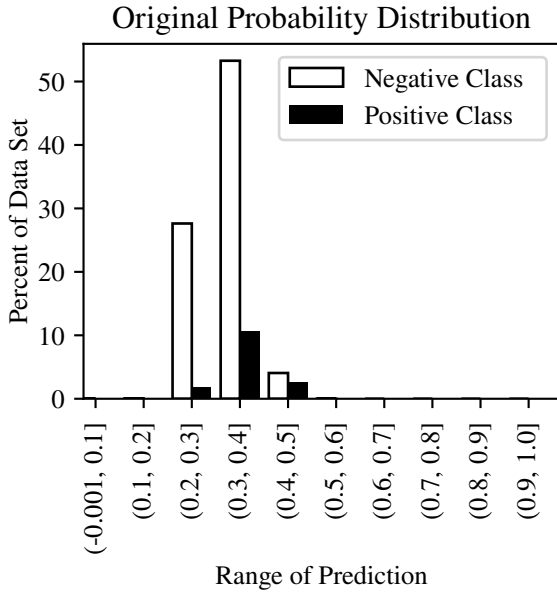
Original Probability Distribution

Transformed Probability Distribution

Prediction		N	P
Actual	N	150,771	0
	P	26,621	0

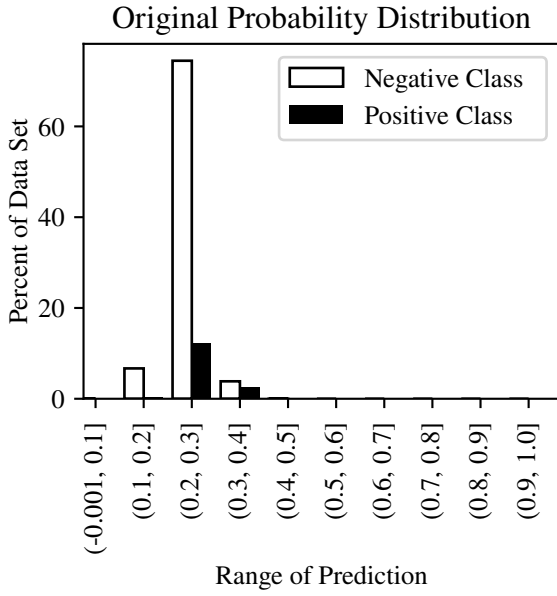| 0.850	Accuracy
0.500	Balanced Accuracy
0.000	Precision
0.000	Balanced Precision
0.000	Recall
0.000	F1
0.000	Balanced F1
0.000	Gmean

Prediction		N	P
Actual	N	111,533	39,238
	P	12,521	14,100

0.708	Accuracy
0.635	Balanced Accuracy
0.264	Precision
0.671	Balanced Precision
0.530	Recall
0.353	F1
0.592	Balanced F1
0.442	Gmean

ROC Curve

4.6. Binary Focal Crossentropy with Class Balancing and $\gamma = 2.0$

Original Probability Distribution

Transformed Probability Distribution

Prediction		N	P
Actual	N	150,771	0
	P	26,621	0

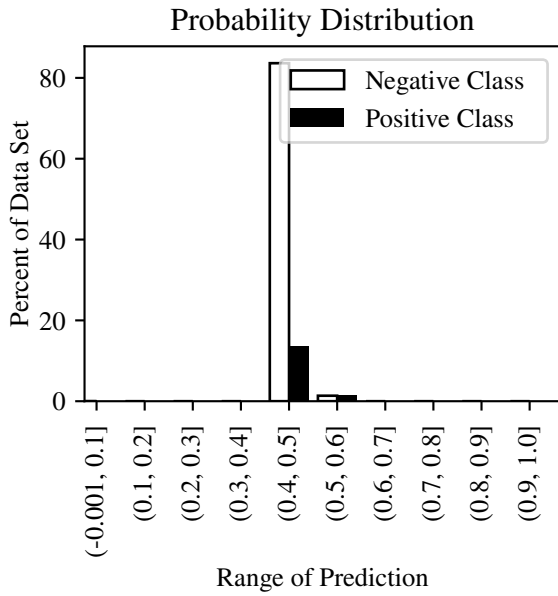| 0.850	Accuracy
0.500	Balanced Accuracy
0.000	Precision
0.000	Balanced Precision
0.000	Recall
0.000	F1
0.000	Balanced F1
0.000	Gmean
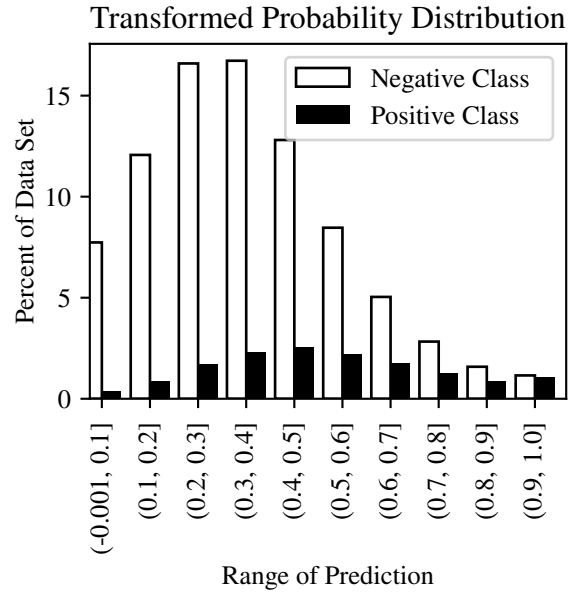
Prediction		N	P
Actual	N	116,943	33,828
	P	13,850	12,771

0.731	Accuracy
0.628	Balanced Accuracy
0.274	Precision
0.681	Balanced Precision
0.480	Recall
0.349	F1
0.563	Balanced F1
0.461	Gmean

For this model we took out the α parameter and set `apply_class_balancing=True`.

ROC Curve

4.7. AdaBoost

Probability Distribution (left)

Probability Distribution

Probability Distribution (right)

Probability Distribution
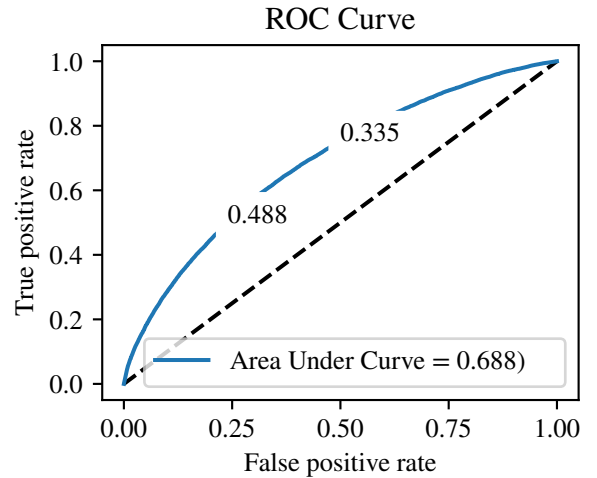
Prediction		N	P
Actual	N	148,358	2,413
	P	23,995	2,626

0.851	Accuracy
0.541	Balanced Accuracy
0.521	Precision
0.860	Balanced Precision
0.099	Recall
0.166	F1
0.177	Balanced F1
0.716	Gmean
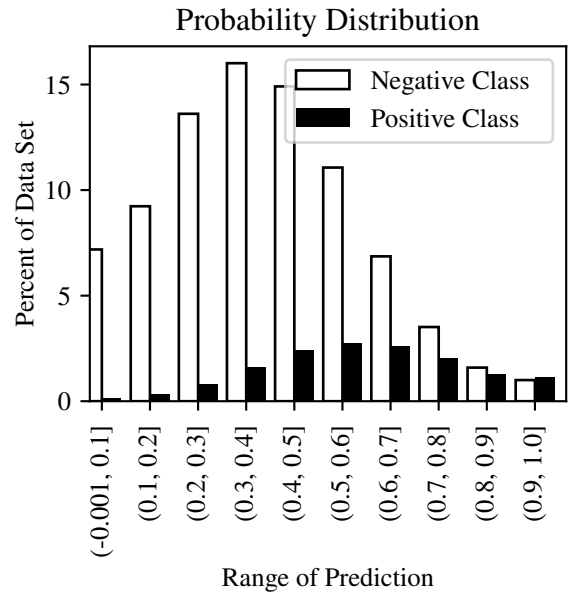
Prediction		N	P
Actual	N	108,133	42,638
	P	9,298	17,323

0.707	Accuracy
0.684	Balanced Accuracy
0.289	Precision
0.697	Balanced Precision
0.651	Recall
0.400	F1
0.673	Balanced F1
0.455	Gmean
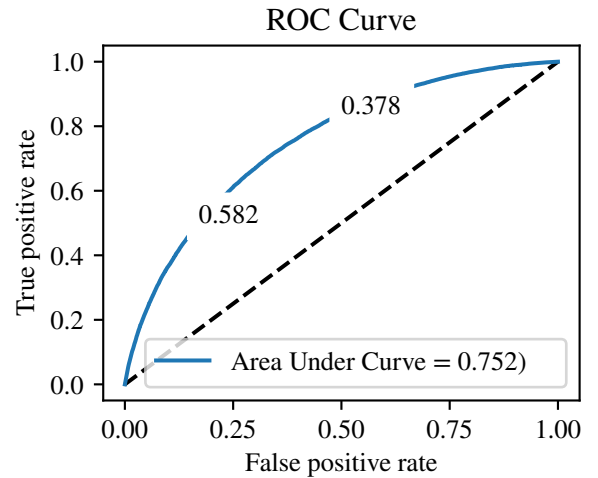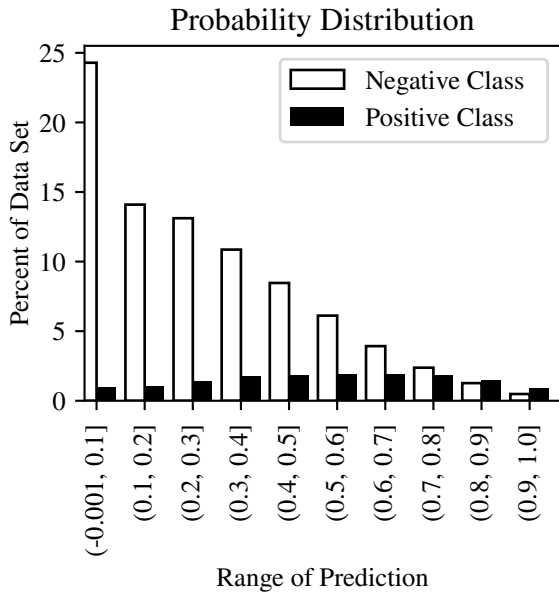
ROC Curve

ROC Curve

Area Under Curve = 0.752)

4.8. Bagging

Probability Distribution

Probability Distribution

Prediction		N	P
Actual	N	125,633	25,138
	P	12,475	14,146

0.788	Accuracy
0.682	Balanced Accuracy
0.360	Precision
0.761	Balanced Precision
0.531	Recall
0.429	F1
0.626	Balanced F1
0.548	Gmean
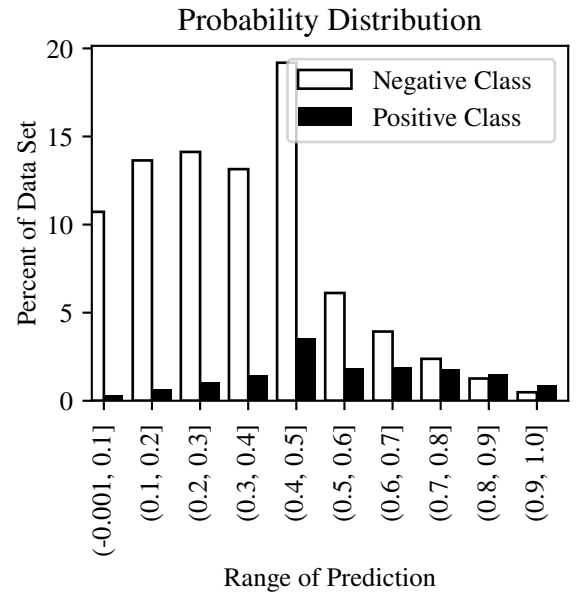
Prediction		N	P
Actual	N	125,633	25,138
	P	12,475	14,146

0.788	Accuracy
0.682	Balanced Accuracy
0.360	Precision
0.761	Balanced Precision
0.531	Recall
0.429	F1
0.626	Balanced F1
0.548	Gmean

Our linear transformation took $p = 0.5$ to itself, so the discrete metrics did not change.

ROC Curve

4.9. Balanced Random Forest Classifier

Probability Distribution

Probability Distribution
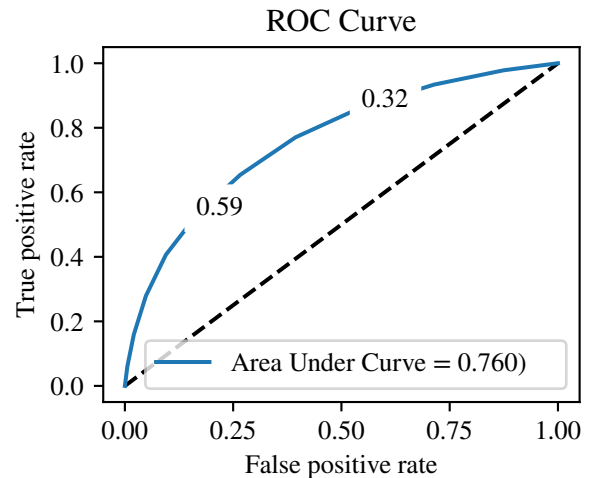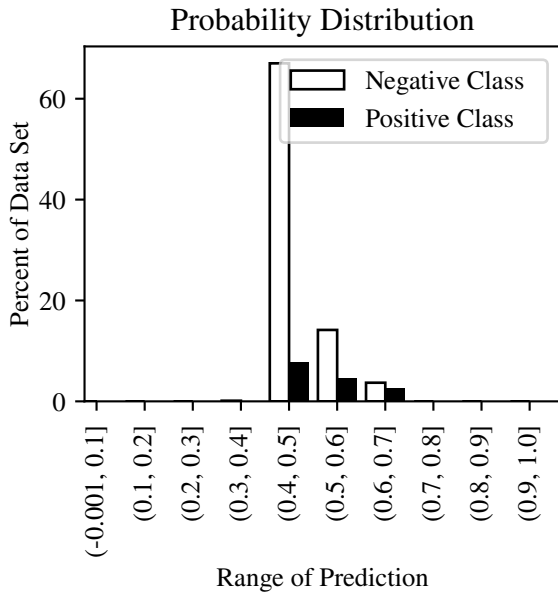
Prediction		N	P
Actual	N	119,102	31,669
	P	13,649	12,972

0.745	Accuracy
0.639	Balanced Accuracy
0.291	Precision
0.699	Balanced Precision
0.487	Recall
0.364	F1
0.574	Balanced F1
0.479	Gmean
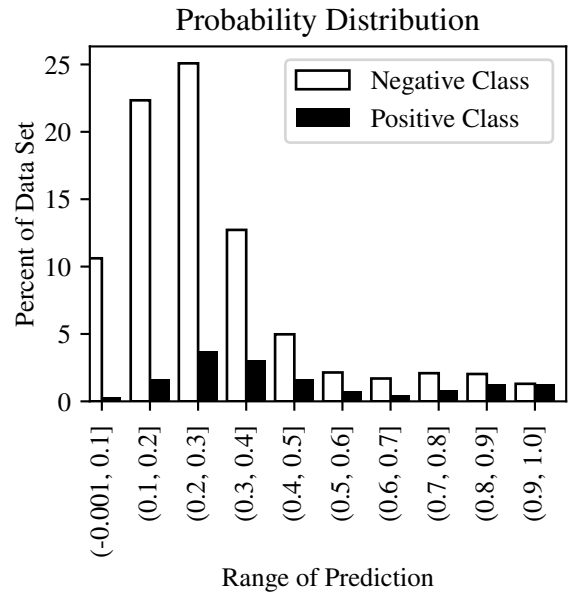
Prediction		N	P
Actual	N	134,347	16,424
	P	18,475	8,146

0.803	Accuracy
0.599	Balanced Accuracy
0.332	Precision
0.737	Balanced Precision
0.306	Recall
0.318	F1
0.433	Balanced F1
0.544	Gmean

ROC Curve
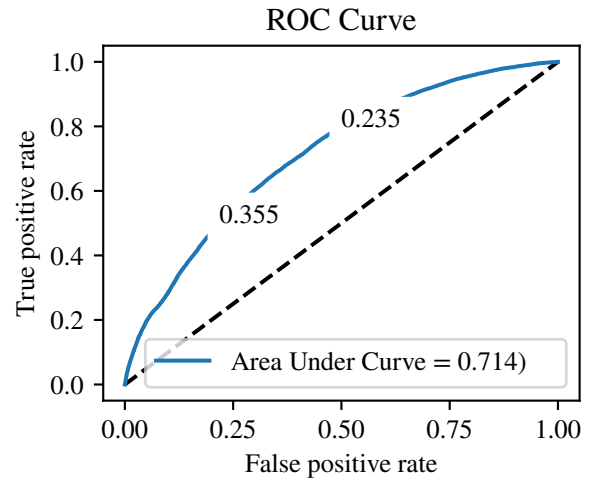
5. Conclusions

6. Discussion

7. 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`

8.

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

Aloul, F., Zualkernan, I., Abu-Salma, R., Al-Ali, H., Al-Merri, M., 2014. ibump: Smartphone application to detect car accidents. 2014 International Conference on Industrial Automation, Information and Communications Technology, Industrial Automation, Information and Communications Technology (IAICT), 2014 International Conference on , 52 – 56URL: `https://ezproxyprod.ucs.louisiana.edu:2443/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid,url&db=edseee&AN=edseee.6922107&site=eds-live&scope=site`.

Basso, F., Pezoa, R., Varas, M., Villalobos, M., 2021. A deep learning approach for real-time crash prediction using vehicle-by-vehicle data. Accident Analysis & Prevention 162, 106409. URL: `https://www.sciencedirect.com/science/article/pii/S0001457521004401`, doi:`https://doi.org/10.1016/j.aap.2021.106409`.

BlinkApp LLC, . Blink! URL: `https://www.blinkapp.net`.

Breiman, L., 1996. Bagging predictors. Machine Learning 24. URL: `https://doi.org/10.1007/BF00058655`.

Cai, Q., Abdel-Aty, M., Yuan, J., Lee, J., Wu, Y., 2020. Real-time crash prediction on expressways using deep generative models. Transportation Research Part C: Emerging Technologies 117, 102697. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X20306124`, doi:`https://doi.org/10.1016/j.trc.2020.102697`.

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

Chen, T., Lu, Y., Fu, X., Sze, N., Ding, H., 2022a. A resampling approach to disaggregate analysis of bus-involved crashes using panel data with excessive zeros. Accident Analysis & Prevention 164, 106496. URL: `https://www.sciencedirect.com/science/article/pii/S0001457521005273`, doi:`https://doi.org/10.1016/j.aap.2021.106496`.

Chen, T., Shi, X., Wong, Y.D., Yu, X., 2020. Predicting lane-changing risk level based on vehicles' space-series features: A pre-emptive learning approach. Transportation Research Part C: Emerging Technologies 116, 102646. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X20305611`, doi:`https://doi.org/10.1016/j.trc.2020.102646`.

Chen, Z., Liu, K., Wang, J., Yamamoto, T., 2022b. H-convlstm-based bagging learning approach for ride-hailing demand prediction considering imbalance problems and sparse uncertainty. Transportation Research Part C: Emerging Technologies 140, 103709. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X22001474`, doi:`https://doi.org/10.1016/j.trc.2022.103709`.

Elamrani Abou Elassad, Z., Mousannif, H., Al Moatassime, H., 2020. A real-time crash prediction fusion framework: An imbalance-aware strategy for collision avoidance systems. Transportation Research Part C: Emerging Technologies 118, 102708. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X20306239`, doi:`https://doi.org/10.1016/j.trc.2020.102708`.

Formosa, N., Quddus, M., Ison, S., Abdel-Aty, M., Yuan, J., 2020. Predicting real-time traffic conflicts using deep learning. Accident Analysis & Prevention 136, 105429. URL: `https://www.sciencedirect.com/science/article/pii/S000145751930973X`, doi:`https://doi.org/10.1016/j.aap.2019.105429`.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55, 119–139. URL: https://www.sciencedirect.com/science/article/pii/S002200009791504X, doi:https://doi.org/10.1006/jcss.1997.1504.

Google, . Get help a!er a car crash (pixel 6 pro). URL: https://partnerdash.google.com/apps/simulator/#get-help-after-a-car-crash?l=en&model=Pixel_6_Pro.

Guo, M., Zhao, X., Yao, Y., Yan, P., Su, Y., Bi, C., Wu, D., 2021. A study of freeway crash risk prediction and interpretation based on risky driving behavior and traffic flow data. Accident Analysis & Prevention 160, 106328. URL: https://www.sciencedirect.com/science/article/pii/S0001457521003596, doi:https://doi.org/10.1016/j.aap.2021.106328.

Haule, H.J., Ali, M.S., Alluri, P., Sando, T., 2021. Evaluating the effect of ramp metering on freeway safety using real-time traffic data. Accident Analysis & Prevention 157, 106181. URL: https://www.sciencedirect.com/science/article/pii/S0001457521002128, doi:https://doi.org/10.1016/j.aap.2021.106181.

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

Islam, Z., Abdel-Aty, M., Cai, Q., Yuan, J., 2021. Crash data augmentation using variational autoencoder. Accident Analysis & Prevention 151, 105950. URL: https://www.sciencedirect.com/science/article/pii/S000145752031770X, doi:https://doi.org/10.1016/j.aap.2020.105950.

Jiang, F., Yuen, K.K.R., Lee, E.W.M., 2020. A long short-term memory-based framework for crash detection on freeways with traffic data of different temporal resolutions. Accident Analysis & Prevention 141, 105520. URL: https://www.sciencedirect.com/science/article/pii/S0001457519317713, doi:https://doi.org/10.1016/j.aap.2020.105520.

Khan, W.A., Ma, H.L., Chung, S.H., Wen, X., 2021. Hierarchical integrated machine learning model for predicting flight departure delays and duration in series. Transportation Research Part C: Emerging Technologies 129, 103225. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21002394, doi:https://doi.org/10.1016/j.trc.2021.103225.

King, G., Zeng, L., 2001. Logistic regression in rare events data. Political Analysis 9, 137 – 163.

Lack, C.D., Berkow, K.S., Gao, Y., 2021. Insights into motor carrier crashes: A preliminary investigation of fmcsa inspection violations. Accident Analysis & Prevention 155, 106105. URL: https://www.sciencedirect.com/science/article/pii/S0001457521001366, doi:https://doi.org/10.1016/j.aap.2021.106105.

Li, P., Abdel-Aty, M., Yuan, J., 2020. Real-time crash risk prediction on arterials based on lstm-cnn. Accident Analysis & Prevention 135, 105371. URL: https://www.sciencedirect.com/science/article/pii/S0001457519311108, doi:https://doi.org/10.1016/j.aap.2019.105371.

Li, Y., Li, M., Yuan, J., Lu, J., Abdel-Aty, M., 2021. Analysis and prediction of intersection traffic violations using automated enforcement system data. Accident Analysis & Prevention 162, 106422. URL: https://www.sciencedirect.com/science/article/pii/S000145752100453X, doi:https://doi.org/10.1016/j.aap.2021.106422.

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.

Lin, Y., Li, L., Jing, H., Ran, B., Sun, D., 2020. Automated traffic incident detection with a smaller dataset based on generative adversarial networks. Accident Analysis & Prevention 144, 105628. URL: https://www.sciencedirect.com/science/article/pii/S0001457519314150, doi:https://doi.org/10.1016/j.aap.2020.105628.

Liu, Y., Lyu, C., Liu, Z., Cao, J., 2021. Exploring a large-scale multi-modal transportation recommendation system. Transportation Research Part C: Emerging Technologies 126, 103070. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21000954, doi:https://doi.org/10.1016/j.trc.2021.103070.

Man, C.K., Quddus, M., Theofilatos, A., 2022. Transfer learning for spatio-temporal transferability of real-time crash prediction models. Accident Analysis & Prevention 165, 106511. URL: https://www.sciencedirect.com/science/article/pii/S000145752100542X, doi:https://doi.org/10.1016/j.aap.2021.106511.

Mohammadi, R., He, Q., Ghofrani, F., Pathak, A., Aref, A., 2019. Exploring the impact of foot-by-foot track geometry on the occurrence of rail defects. Transportation Research Part C: Emerging Technologies 102, 153–172. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18314001, doi:https://doi.org/10.1016/j.trc.2019.03.004.

Morris, C., Yang, J.J., 2021. Effectiveness of resampling methods in coping with imbalanced crash data: Crash type analysis and predictive modeling. Accident Analysis & Prevention 159, 106240. URL: https://www.sciencedirect.com/science/article/pii/S0001457521002712, doi:https://doi.org/10.1016/j.aap.2021.106240.

National Center for Statistics and Analysis, 2022. Crash Report Sampling System analytical user's manual, 2016-2020. Technical Report DOT HS 813 236. National Highway Traffic Safety Administration.

NHTSA, 2016-2020. Crash report sampling system. https://www.nhtsa.gov/crash-data-systems/crash-report-sampling-system.

Orsini, F., Gecchele, G., Rossi, R., Gastaldi, M., 2021. A conflict-based approach for real-time road safety analysis: Comparative evaluation with crash-based models. Accident Analysis & Prevention 161, 106382. URL: https://www.sciencedirect.com/science/article/pii/S0001457521004139, doi:https://doi.org/10.1016/j.aap.2021.106382.

Park, H., Oh, C., 2019. A vehicle speed harmonization strategy for minimizing inter-vehicle crash risks. Accident Analysis & Prevention 128, 230–239. URL: https://www.sciencedirect.com/science/article/pii/S0001457519300314, doi:https://doi.org/10.1016/j.aap.2019.04.014.

Parsa, A.B., Taghipour, H., Derrible, S., Mohammadian, A.K., 2019. Real-time accident detection: Coping with imbalanced data. Accident Analysis & Prevention 129, 202–210. URL: https://www.sciencedirect.com/science/article/pii/S0001457519301642, doi:https://doi.org/10.1016/j.aap.2019.05.014.

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

Peng, Y., Li, C., Wang, K., Gao, Z., Yu, R., 2020. Examining imbalanced classification algorithms in predicting real-time traffic crash risk. Accident Analysis & Prevention 144, 105610. URL: https://www.sciencedirect.com/science/article/pii/S0001457519306906, doi:https://doi.org/10.1016/j.aap.2020.105610.

Rahman, M., 2019. Google accidentally rolls out "personal safety" app, confirming car crash detection is coming to the pixel. URL: https://www.xda-developers.com/google-pixel-car-crash-detection/.

Schlögl, M., 2020. A multivariate analysis of environmental effects on road accident occurrence using a balanced bagging approach. Accident Analysis & Prevention 136, 105398. URL: https://www.sciencedirect.com/science/article/pii/S0001457519308516, doi:https://doi.org/10.1016/j.aap.2019.105398.

Schlögl, M., Stütz, R., Laaha, G., Melcher, M., 2019. A comparison of statistical learning methods for deriving determining factors of accident occurrence from an imbalanced high resolution dataset. Accident Analysis & Prevention 127, 134–149. URL: https://www.sciencedirect.com/science/article/pii/S0001457518307760, doi:https://doi.org/10.1016/j.aap.2019.02.008.

Shi, Q., Zhang, H., 2021. An improved learning-based lstm approach for lane change intention prediction subject to imbalanced data. Transportation Research Part C: Emerging Technologies 133, 103414. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21004083, doi:https://doi.org/10.1016/j.trc.2021.103414.

Shi, X., Wong, Y.D., Li, M.Z.F., Palanisamy, C., Chai, C., 2019. A feature learning approach based on xgboost for driving assessment and risk prediction. Accident Analysis & Prevention 129, 170–179. URL: https://www.sciencedirect.com/science/article/pii/S0001457518310820, doi:https://doi.org/10.1016/j.aap.2019.05.005.

Sosmart SAP, . SOSmart automatic car crash detection app. URL: http://www.sosmartapp.com.

Tomek, I., 1976. Two modifications of cnn. IEEE transactions on Systems, Man and Communications, SMC 6, 769–772.

White, J., Thompson, C., Turner, H., Dougherty, B., Schmidt, D.C., 2011. Wreckwatch: Automatic traffic accident detection and notification with smartphones. MOBILE NETWORKS AND APPLICATIONS 16, 285 – 303. URL: https://ezproxyprod.ucs.louisiana.edu:2443/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid,url&db=edsbl&AN=RN290973892&site=eds-live&scope=site.

Winkler, R., 2021. Apple wants iphones to detect car crashes, auto-dial 911. The Wall Street Journal Eastern Edition , 1URL: https://ezproxyprod.ucs.louisiana.edu:2443/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid,url&db=edsgao&AN=edsgcl.681029711&site=eds-live&scope=site.

Yahaya, M., Guo, R., Fan, W., Bashir, K., Fan, Y., Xu, S., Jiang, X., 2021a. Bayesian networks for imbalance data to investigate the contributing factors to fatal injury crashes on the ghanaian highways. Accident Analysis & Prevention 150, 105936. URL: https://www.sciencedirect.com/science/article/pii/S0001457520317565, doi:https://doi.org/10.1016/j.aap.2020.105936.

Yahaya, M., Guo, R., Jiang, X., Bashir, K., Matara, C., Xu, S., 2021b. Ensemble-based model selection for imbalanced data to investigate the contributing factors to multiple fatality road crashes in ghana. Accident Analysis & Prevention 151, 105851. URL: https://www.sciencedirect.com/science/article/pii/S0001457520316717, doi:https://doi.org/10.1016/j.aap.2020.105851.

Yu, R., Wang, Y., Zou, Z., Wang, L., 2020. Convolutional neural networks with refined loss functions for the real-time crash risk analysis. Transportation Research Part C: Emerging Technologies 119, 102740. URL: https://www.sciencedirect.com/science/article/pii/S0968090X20306549, doi:https://doi.org/10.1016/j.trc.2020.102740.

Zhu, S., Wan, J., 2021. Cost-sensitive learning for semi-supervised hit-and-run analysis. Accident Analysis & Prevention 158, 106199. URL: https://www.sciencedirect.com/science/article/pii/S000145752100230X, doi:https://doi.org/10.1016/j.aap.2021.106199.