Predicting Severity of Traffic Accidents in Philadelphia, PA
CIS519 Project, 5 May 2020

Leanne Chan LLCHAN@DESIGN.UPENN.EDU
Yangyi Shi YANGYIS@SEAS.UPENN.EDU
Xinran Liu XRLIU@SEAS.UPENN.EDU

Abstract

Traffic accidents are costly. This project identifies accident hot-spots based on current data and runs a predictive model to predict under which circumstances an accident would occur in these hot-spots, and if so, the severity of the accident. It then compares the performance of different algorithms for multi-classification such as SVM, random forest, and multinomial logistic regression. Finally, suggestions for practical implementations of the model are discussed.

1. Introduction

1.1. Motivation and Aim

According to a report by Penn Live, 1,200 people died in traffic accidents in PA in 2018 alone. Given the cost of human life, and potential structural damage, identifying areas with a high likelihood of an accident is necessary to target measures to reduce such incidents. Most public agencies currently use past accident frequency to target their policies. However, such an approach does not take into account the dynamic nature of environmental and structural factors such as weather and time. An area with a high frequency of past accidents is not a hot-spot at all times of a day. An accident severity prediction model will allow agencies to deploy spatially and temporally targeted safety measures to hot-spots only when they are predicted to be 'activated'. This project thus aims to create a model that can predict at any time if a hot-spot is 'activated' (predicted to have an accident) and if so, the severity of an accident.

1.2. Dataset

The models were trained on the US Traffic Accidents 2016-2019 (S.Moosavi & R.Ramnath, 2019a)(S.Moosavi & R.Ramnath, 2019b) dataset that contained traffic accident data from February 2016 to December 2019 across the US. We decided to focus on the city of Philadelphia in Pennsylvania, where there were a total of 6224 accidents.

Table 1. Results of DBSCAN Clustering Algorithm

MAX BOUND	MIN SAMPLES	CLUSTERS	POINTS
20	15	86	3354
25	15	85	3403
20	20	62	2921
25	20	64	2997
20	25	48	2602

Other than location data, the dataset contained temporal data, weather data and road design of each accident. Additionally, each accident had a severity score of 1-4 which was used as the dependent variable in the model.

Demographic data such as median income for each area was also obtained from the US Census and added to each instance using the latitude and longitude.

2. Data Processing

Data processing methodology was largely taken from a Machine Learning project to predict traffic accidentses in London, UK (Antonio, 2019).

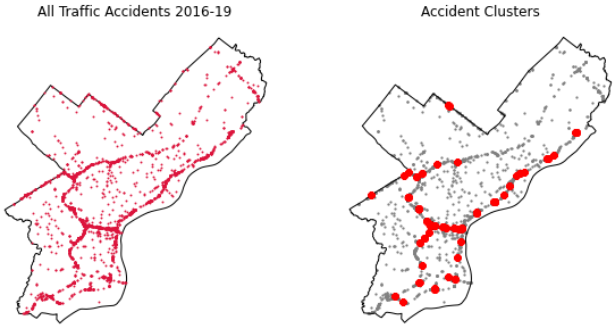
2.1. Cluster Analysis

To identify accident hot-spots, the DBSCAN clustering algorithm was run on the 6224 accident points in Philadelphia using different parameters. The parameter max bound indicates the maximum distance between two points in the cluster and the min samples is the minimum number of points to be considered a cluster. The results in Table 2 show the number of clusters (hot-spots) identified and total number of points that fall in any cluster. A max bound of 20 meters and min samples of 20 points per clusters was chosen, resulting in 62 areas identified as hot-spots.

During clustering, each point in the original dataset is assigned a label identifying the cluster it belongs to. Each accident instance represents a different feature setting for each cluster under which an accident occurred, together with an associated severity score. Each of these points rep-

resent a situation in which the hot-spot is 'activated' or in which the severity score was non-zero. The clusters and labels allow for prediction on targeted areas rather than on an infinite number of points.

Figure 1. Traffic Accidents and Clusters in Philadelphia



2.2. Negative Sampling

For each accident in a cluster, 3 points were randomly generated in the cluster. It was ensured that no point in cluster had the exact same features as a positive sample. The severity scores of these negative samples were set to zero.
After the data processing, each instance in the final dataset had environmental data, a severity score (0 if no accident), which would be the dependent variable, and an associated cluster, which would allow future predictions to be made on clusters.

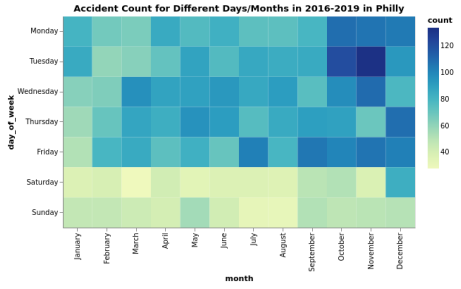
3. Exploratory Data Analysis

The accident dataset was filtered to Philadelphia and relationships between features were explored.

3.1. Temporal Relationships

Figure 2 indicates a macro trend over months and days. There is a clear increase in the number of accidents during the winter months and on the weekdays, which is likely due to workers commuting.

Figure 2. Count of Traffic Accidents by Month and Day



A similar heatmap was plotted to analyze hourly trends. The heatmap confirmed that the highest accident counts occur during the morning (6-8am) and evening (4-6pm) rush hours.

Figure 3. Count of Traffic Accidents by Day and Time

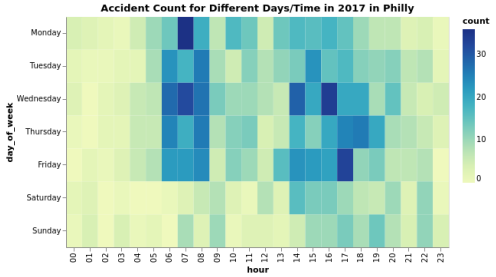
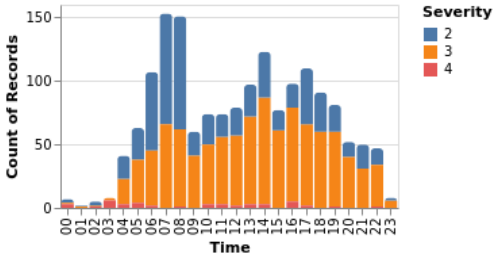


Figure 4 indicates the severity of accidents over a day. There is a clear class imbalance with a majority of the accidents being classified as '3' for the year 2019.

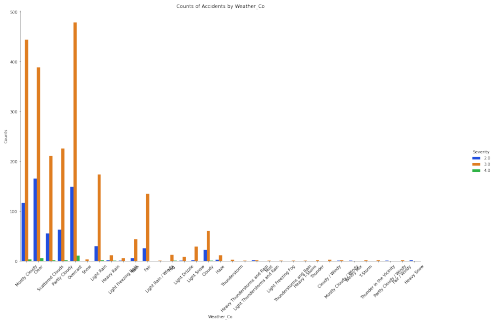
Figure 4. Severity of Accidents per hour in 2019



3.2. Environmental Relationships

Figure 5 showed the count of accidents by severity and weather condition. There were many options for weather condition, and most accidents occurred only at a few types of conditions. Though it was not done in this project, future version of this project could include feature engineering of weather to reflect this distribution.

Figure 5. Severity by Weather Conditions



4. Model Training and Validation

4.1. Model

The dependent variable in the model was the severity score of accidents (0,2,3,4), with 4 being the highest severity and 2 the lowest. 0 indicated no accident. Features chosen for the model were weather characteristics such as precipitation and humidity, wind direction, month, and time of day.

Each accident instance had an associated cluster for future identification if a cluster was 'activated (severity>0)' or 'not activated (severity=0)'.

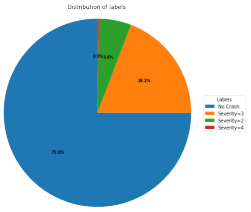
4.2. Class Imbalance

As a result of the negative sampling and a reflection of the natural state of the world, there were more 'non-accidents' than 'accidents'. This resulted in a huge class imbalance as in figure 6. Within the 'accidents', there was also am imbalance amongst severity, with there being very few of highest severity. The following methods were used to treat the class imbalance resulting in skewed accuracy.

4.2.1. Class Weights

Class weights were assigned when running multinomial logistic regression and decision trees with adaptive boosting (AdaBoost). Classes assigned a higher weight have more emphasis in the model training.

Figure 6. Distribution of Labels



4.2.2. Upsampling

Upsampling was done on the test set before running logistic regression. This was achieved using Sklearn's resampling function to bring the number of samples for classes 2,3 and 4 equal to the size of class 0.

Another upsampling technique, imblearn's SMOTE (Synthetic Minority Over Sampling Technique) was used before running the voting classifier. All not-majority classes were upsampled.

4.2.3. Undersampling

The opposite to upsampling, undersampling takes a sample from the majority class (class=0 in this model) to be equal to the size of the minority classes. This was used in imblearn's BalanceBagging and easy ensemble classifier, as well as in Sklearn's RandomForest classifier.

4.3. Learning Algorithms

Besides the following models, Support Vector Machines were also tested, but they did not converge.

4.3.1. Multinomial Logistic Regression

Logistic regression is a linear classification model that calculates for each instance the probabilities that the instance is in each of the classes. Since this was a multiclassification problem, the class with the highest probability for that instance was selected. Probability predictions were made using the softmax function.

For severity c=0,2,3,4:

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4) = \frac{exp(\boldsymbol{\theta}_c^T \boldsymbol{x})}{\sum_{c=0,2,3,4} exp(\boldsymbol{\theta}_c^T \boldsymbol{x})}$$

Where x are the bias term and predictors in the model, and $\boldsymbol{\theta}$ are the coefficients of the linear combination optimised by the loss function.

The algorithm optimised the loss function with L1 regularization to avoid over-fitting because of the large set of one-hot encoded features.

$$L_{reg}(\boldsymbol{\theta}_c) = -\sum_{i=1}^{n}[y_i log h_{\boldsymbol{\theta}_c}(x_i) + (1 - y_i) log(1 - h_{\boldsymbol{\theta}_c}(x_i)] + \sum_{j=1}^{d} |\theta_{cj}|$$

The above loss function was minimised using stochastic average gradient descent.

4.3.2. Random Forest

To prevent overfitting and reduce the effect of noisy features, the Random Forest classifier was also used. This approach used an ensemble of decision trees with bootstrapping replication. Bootstrapping replication is the process of constructing new training sets by sampling different instances with replacement, allowing trees to be trained on different subsets of the data. At each split within each tree, a random subset of features were selected as candidates for the split, and the feature that resulted in the highest information gain was selected for the split. The performance of each tree was estimated using out-of-bootstrap data. The prediction of the Random Forest was taken as the mean prediction of the trees.

4.3.3. AdaBoosted Decision Trees

Besides the Random Forest classifier, AdaBoost was used to induce diversity into the model as well. The AdaBoost meta-classifier used with decision trees creates an ensemble of decision trees by repeatedly emphasizing mispredicted instances. This is done by assigning each instance a weight which is updated using a function of the weighted error over all instances in the model at time t. After T iterations, the prediction is made using the aggregated predictions of each of the T decision trees, weighted by a function of their training error.

4.3.4. Alternative Ensembles

Alternative ensembles were also tested, these included the voting classifier, easy ensemble and balanced bagging. Both easy ensembles (similar to random forest) and balanced bagging (similar to AdaBoost) were ensembles from the imblearn package which had an inbuilt parameter to execute undersampling. The voting classifier used the follow-

Table 2. Results of Algorithms

ALGORITHM	TOT ACC	AVG ACC	FALSE NEG
LOGREG	0.203	0.455	0.0
ADABOOST	0.575	0.507	0.291
RF	0.813	0.542	0.560
VOTING	0.753	0.411	0.687
BB	0.361	0.547	0.077
EE	0.345	0.578	0.085

ing classifiers in the ensemble - random forest, extra tree, k-nearest neighbors and support vector machines.

4.4. Parameter Tuning and Result Metric

Sklearn's grid-search functionality was used to select a set of parameters for each model that optimised the objective function. The grid-search cross validation was run on the 70% training set over 3 folds.

As our dataset was extremely imbalanced, the total accuracy was not the most meaningful metric. A model that predicted 0 for all instances would still achieve an accuracy of 75%. Additionally, given the context of traffic accident prediction, the cost of a false negative is more than a false positive. If a cluster was predicted to not have an accident and nothing was done to step up policing or increase stop signs, but an accident did happen, this would be more costly than if interventions were made even though there was not going to be an accident. Hence, the average accuracy over classes and false positive rate were reported for each model. The false negative rate was calculated as the ratio of true 2,3,4 instances wrongly classified as 0, since severity=0 indicated 'no accident'. For a visual indicator of the errors in our model, the confusion matrix was plotted after each model was tested.

5. Results and Analysis

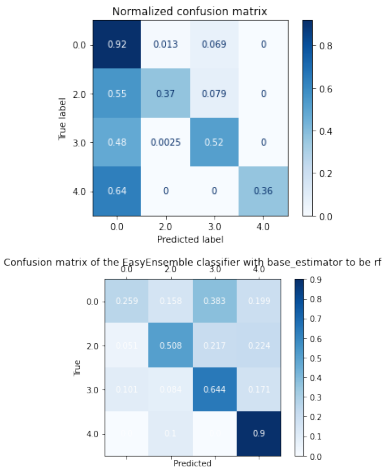
5.1. Model Results

Various models achieved high accuracy in different classes. Although the total accuracy is high for Random Forest and the voting classifier, they have a high false negative rate, as most of the predictions are of the class=0 (75% of the dataset), even after fixing class imbalance. Conversely, the balanced bagging and easy ensemble classifiers have a low total accuracy, but provide the best trade off between average accuracy and false negative rate, thus are the best models amongst this for predicting the severity of accidents.

5.2. Misclassifications

The confusion matrix for Random Forest (best total accuracy) and Easy Ensemble Classifier (good average accuracy and false negative rate) in figure 7 show the misclassifcation rates for each class. Although the random forest classifier has the highest accuracy, it is not good at predicting instances when there are accidents and the severity of the accident. The easy ensemble classifier does better at correctly predicting when an accident does occur, as evidenced by the darker diagonal in the confusion matrix. However, the accuracy for is very poor for when there are no accidents, and half of those instances were predicted as severity=3.

Figure 7. Confusion Matrices



6. Conclusion and Future Work

In theory, the models trained can be used to predict at a given time and environment (e.g. rainy Monday, 7am, December) if an accident will occur at each cluster location in Philadelphia, and if so, the severity of the accident. An application can be built with an interactive map showing the various clusters generated in figure 1. If the cluster is lit up, it indicates that an accident is predicted in that area at that time. The color of the cluster can indicate the severity predicted (2,3 or 4). This application can be used by traffic police to target their policing to enforce safer driving. However, the models presented still suffer from a high false positive rate, despite a good false negative rate. Hence it would still be costly to implement such models in practice.

To improve the model performance, better feature engineering can be done as there were many one-hot encoded predictors in our model. Reducing the number of predictors in the model could reduce the variance in the model. Additionally, since the easy ensemble has a good false negative rate, but misclassifies 0s as 3s, perhaps adjustments can be made to address this, such as adding weights for class 0 in the decision tree. Finally, the problem could also be segmented into a binary classification model of 'accident' vs 'no-accident' and a secondary model to predict the severity of an accident.

Acknowledgments

Dr Eric Eaton, Dr Dinesh Jayaraman and CIS519 TAs.

References

Antonio, M. Live prediction of traffic accident risks using machine learning and google maps, 2019.

S.Moosavi, M.H.Samavatian, S.Parthasarathy and R.Ramnath. A countrywide traffic accident dataset., 2019a.

S.Moosavi, M.H.Samavatian, S.Parthasarathy R.Teodorescu and R.Ramnath. Accident risk prediction based on heterogeneous sparse data: New dataset and insights. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM*, 2019b.