Classification of Injuries Sustained by Non-Motorists

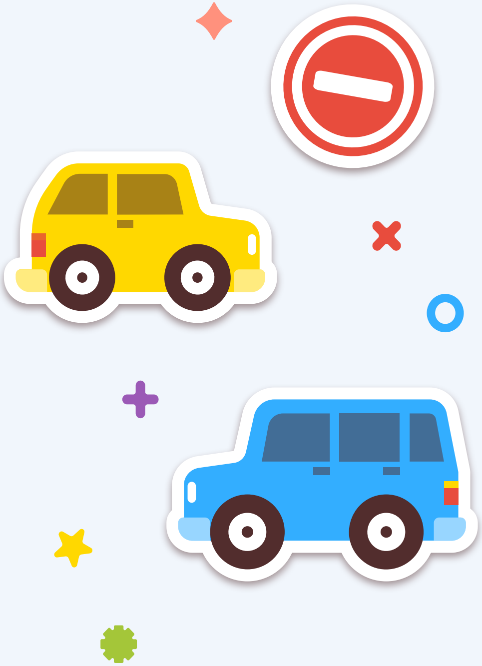
By: Kaavya Borra & Radin Rezanezhad

Table of contents

Table of contents

01

Introduction



Did you know?

Every year, the U.S. experiences more than **6 million** automobile accidents involving passenger vehicles. Traffic collisions rank as the number one cause of fatalities in the country, with yearly death tolls surpassing **38,000**.

Problem

- Current police dispatch systems rely mainly on verbal reports
- Emergency responders often lack critical information about injury severity before arriving on scene
- Resource allocation decisions must be made promptly with limited data
    - Incorrect severity assessments could lead to insufficient emergency responses, which could worsen the situation

Why It Matters

- Non-motorists (pedestrians/cyclists) are very vulnerable to severe injuries
- Response time and proper resource allocation are critical for survival
- Each minute saved in emergency response can significantly impact patient outcomes

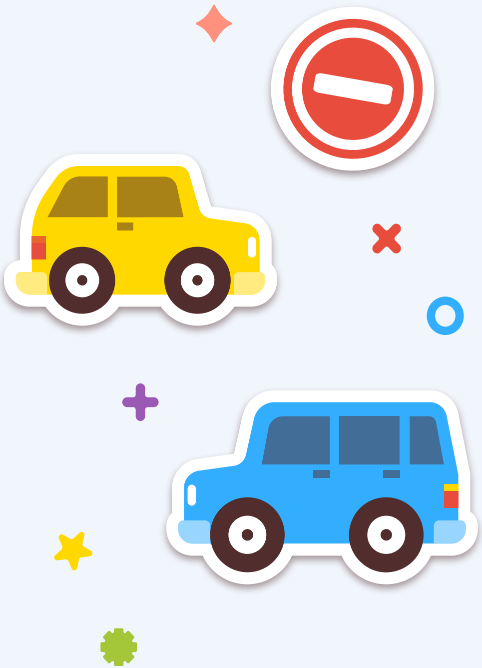
Project Goals

Develop a classification model to predict non-motorist **injury severity** in vehicle collisions using data:

- Collision characteristics and type
- Temporal patterns and time of day
- Environmental and weather conditions
- Specific location attributes and road features

02

Description of Dataset



Scope and Source

Focused on county and local roadways in Montgomery County

Data collected through Maryland State Police's Automated Crash Reporting System (ACRS)

Contained 6,104 unique collision instances

Total of 28 distinct attributes per instance

Distribution of Injury Severity (class)

Suspected Minor Injury	45%
Possible Injury	32%
Suspected Serious Injury	10%
No Apparent Injury	10%
Fatal Injury	2%

Data Considerations

- 12,610 blank values across all attributes
    - Some attributes were mostly blank
- Hidden missing values

03

Pre-Processing

- Done on Google Sheets
- Used Weka for visualization

1

Columns with over 75% missing values or irrelevant/redundant columns were removed

- Report IDs and case identifiers have nothing to do with the crash
- Report Number, Local Case Number, ACRS Report Type, Cross Street Names, Road Names, Municipality, Person ID, Pedestrian Type

2

- Split column in excel to separate date and time
  - Crash Date: Days past Jan 1, 1990
  - Crash Time: Hours past midnight

3

Combine matching data values into a coherent value for uniformity

- Replace values like “SNOW” with “Snow”
- Replace values like “Montgomery County Police” with “MONTGOMERY”
- Agency Name, Route Type, Related Non-Motorist, Weather, Surface Conditions, Light, Traffic Control, Pedestrian Type, Injury Severity

4

Removed all instances of hidden missing values

- "Unknown" and "N/A," Related Non-Motorist, Collision Type, Traffic Control

5

Grouped similar values

- “Fog, Smog, Smoke” was grouped with “FOGGY”
- “MC/BIKE HELMET” became “Helmet”
- “ALCOHOL PRESENT, None Detected,” was grouped with “None Detected”
- Weather, Light, Safety Equipment, Pedestrian Actions, Pedestrian Location, Non-motorist substance abuse, Driver Substance Abuse

6

Split Safety Equipment into whether safety equipment was present or not

- None = “No”
- Everything else = “Yes”

7

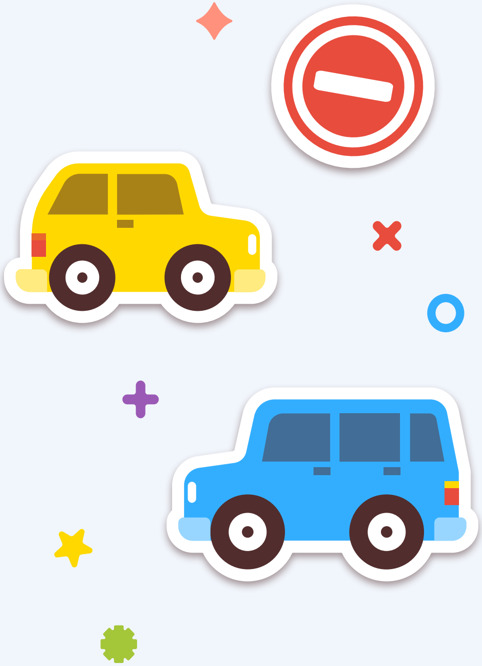
Min/Max Normalization of Latitude and Longitude

- Values were all concentrated in a small area on a big scale

04

Attribute Selection

- Done on WEKA



InfoGainAttributeEval

- Evaluates attributes based on information gain
    - Measures the reduction in entropy caused by partitioning the dataset according to the attribute
- Higher information gain means the attribute provides a better split, reducing more uncertainty
- Used the Ranker search method with a cutoff threshold of 0.015

InfoGainAttributeEval

Attribute selection output

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 20 Injury Severity):
        Information Gain Ranking Filter

Ranked attributes:
 0.03645  15 Pedestrian Location
 0.02572  14 Pedestrian Actions
 0.02099  13 Pedestrian Movement
 0.01982   5 Related Non-Motorist
 0.01677   4 Route Type
 0.01545   3 Hours past Midnight
 0.01525   6 Collision Type
 0.01422   9 Light
 0.0135   12 Non-Motorist Substance Abuse
 0.0114   10 Traffic Control
 0.01018   1 Agency Name
 0.00772  16 At Fault
 0.00677  11 Driver Substance Abuse
 0.00648   7 Weather
 0.00332   8 Surface Condition
 0.00332  17 Safety Equipment
 0        18 Lat- Normalized
 0         2 Days after Jan 1
 0        19 Long- Normalized

Selected attributes: 15,14,13,5,4,3,6,9,12,10,1,16,11,7,8,17,18,2,19 : 19
```

GainRatioAttributeEval

- Evaluates attributes based on the gain ratio
  - Similar to InfoGainAttributeEval but it accounts for the number and size of branches when choosing an attribute
- Gain ratio will favor attributes that not only provide a good split (high InfoGain) but also distribute the dataset into meaningful subsets (low SplitInfo)
- Used the Ranker search method with a cutoff threshold of 0.1100

GainRatioAttributeEval

Attribute selection output

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 20 Injury Severity):
        Gain Ratio feature evaluator

Ranked attributes:
 0.05316  11 Driver Substance Abuse
 0.04505  12 Non-Motorist Substance Abuse
 0.02156   3 Hours past Midnight
 0.01765   5 Related Non-Motorist
 0.01432  14 Pedestrian Actions
 0.01354  15 Pedestrian Location
 0.01233   6 Collision Type
 0.01149   1 Agency Name
 0.01087   9 Light
 0.01076   4 Route Type
 0.0101   13 Pedestrian Movement
 0.0093   16 At Fault
 0.00727  10 Traffic Control
 0.00622   7 Weather
 0.00597  17 Safety Equipment
 0.00563   8 Surface Condition
 0        18 Lat- Normalized
 0         2 Days after Jan 1
 0        19 Long- Normalized

Selected attributes: 11,12,3,5,14,15,6,1,9,4,13,16,10,7,17,8,18,2,19 : 19
```

OneRAttributeEval

- Evaluates attributes using the OneR classifier
  - Creates one rule for each attribute and selects the rule with the smallest error rate
- Used the Ranker search method with a cutoff threshold of 44.5

OneRAttributeEval

Attribute selection output

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 20 Injury Severity):
        OneR feature evaluator.

        Using 10 fold cross validation for evaluating attributes.
        Minimum bucket size for OneR: 6

Ranked attributes:
44.954  15 Pedestrian Location
44.594  10 Traffic Control
44.577   8 Surface Condition
44.561   7 Weather
44.545   9 Light
44.545   1 Agency Name
44.545  17 Safety Equipment
44.545  16 At Fault
44.463   4 Route Type
44.463   5 Related Non-Motorist
44.397  13 Pedestrian Movement
44.397  14 Pedestrian Actions
44.397  11 Driver Substance Abuse
44.332  12 Non-Motorist Substance Abuse
44.282   6 Collision Type
40.908  18 Lat- Normalized
40.809  19 Long- Normalized
40.76    2 Days after Jan 1
40.367   3 Hours past Midnight

Selected attributes: 15,10,8,7,9,1,17,16,4,5,13,14,11,12,6,18,19,2,3 : 19
```

ReliefFAttributeEval

- Evaluates attributes by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class
    - Effective for detecting conditional dependencies between attributes
- Higher weights indicate more important features for distinguishing between classes
- Used the Ranker search method with a cutoff threshold of 0.009

ReliefFAttributeEval

Attribute selection output

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 20 Injury Severity):
        ReliefF Ranking Filter
        Instances sampled: all
        Number of nearest neighbours (k): 10
        Equal influence nearest neighbours

Ranked attributes:
 0.027354  15 Pedestrian Location
 0.018023  13 Pedestrian Movement
 0.016565  10 Traffic Control
 0.013341   4 Route Type
 0.010529  14 Pedestrian Actions
 0.009833   6 Collision Type
 0.009529   9 Light
 0.008194   5 Related Non-Motorist
 0.006928   2 Days after Jan 1
 0.006194   1 Agency Name
 0.005893   7 Weather
 0.00454   16 At Fault
 0.004455   3 Hours past Midnight
 0.003507  17 Safety Equipment
 0.002649   8 Surface Condition
 0.001727  18 Lat- Normalized
 0.000804  19 Long- Normalized
 0.000581  12 Non-Motorist Substance Abuse
 0.000338  11 Driver Substance Abuse

Selected attributes: 15,13,10,4,14,6,9,5,2,1,7,16,3,17,8,18,19,12,11 : 19
```

Our Pick

- Chose the attributes which were picked by 2 or more of the above algorithms
    - Pedestrian Location, Pedestrian Actions, Collision Type, Related Non-Motorist, Route Type, Hours Past Midnight, Agency Name, Pedestrian Movement, Traffic Control, and Light

05

Model Classifiers

NaiveBayes

- Uses probability theory to classify data based on Bayes' theorem, assuming that the attributes are independent of each other
- Calculates the likelihood of each class given the attributes and selects the class with the highest probability

J48

- Generates a decision tree by recursively splitting the dataset into subsets based on the attribute that provides the highest information gain at each step
    - Each internal node in the tree represents a test on an attribute, and the branches represent the outcomes of that test
    - Final leaf nodes indicate the predicted class

RandomForest

- Builds an ensemble of decision trees, where each tree is constructed by splitting data based on randomly chosen attributes
  - Each internal node in a tree represents a test on one of these attributes, and the branches indicate the possible outcomes of the test
  - Continues until a leaf node is reached, which holds the predicted class

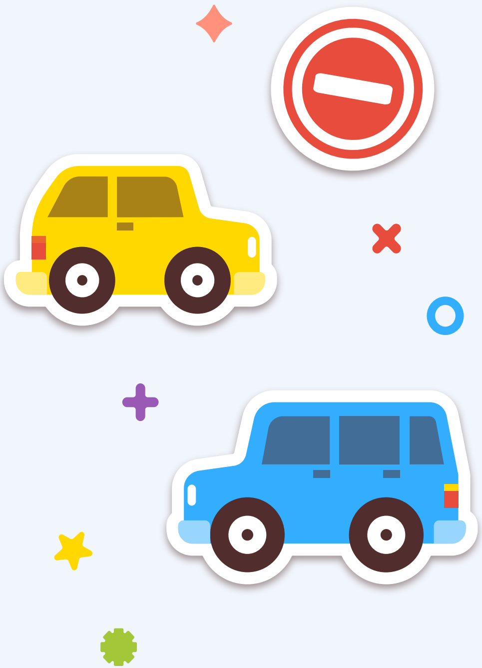
KStar

- Classifies new data points by comparing them to existing examples in the dataset using an entropy-based distance measure
  - Identifies the most similar neighbors and assigns a class based on these closest examples for each new instance
- Calculates the probability of transforming one instance into another through a series of attribute changes, which serves as the similarity function

06

Results

- Done on WEKA



NaiveBayes

	InfoGain	GainRatio	OneR	ReliefF	Our Pick
Accuracy	40.3846%	41.5929%	43.4426%	40.3846%	39.4231%
TP Rate	0.404	0.416	0.434	0.404	0.394
FP Rate	0.391	0.449	0.455	0.387	0.392
ROC	0.532	0.568	0.546	0.525	0.551

J48

	InfoGain	GainRatio	OneR	ReliefF	Our Pick
Accuracy	39.4231%	42.4779%	45.9016%	41.3462%	39.4231%
TP Rate	0.394	0.425	0.459	0.413	0.394
FP Rate	0.388	0.452	0.459	0.413	0.400
ROC	0.572	0.553	0.500	0.500	0.581

RandomForest

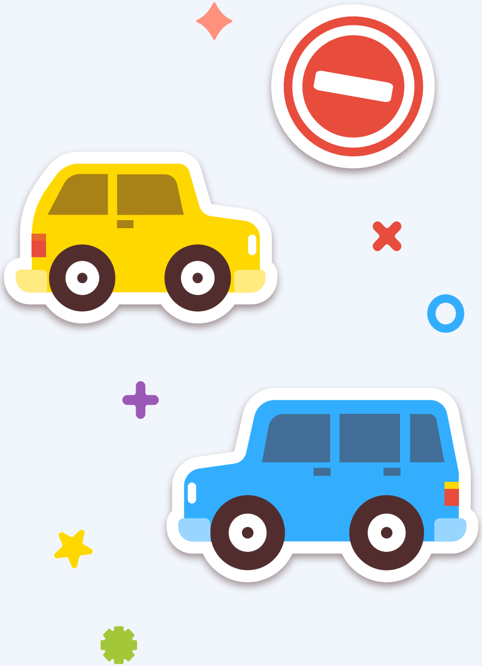
	InfoGain	GainRatio	OneR	ReliefF	Our Pick
Accuracy	26.9231%	37.1681%	45.9016%	33.654%	37.5%
TP Rate	0.269	0.372	0.459	0.337	0.375
FP Rate	0.353	0.353	0.354	0.384	0.307
ROC	0.543	0.561	0.631	0.548	0.568

KStar

	InfoGain	GainRatio	OneR	ReliefF	Our Pick
Accuracy	32.6932%	39.823%	47.541%	41.3462%	41.3462
TP Rate	0.327	0.398	0.475	0.413	0.413
FP Rate	0.402	0.451	0.420	0.386	0.341
ROC	0.517	0.563	0.620	0.549	0.553

07

Analysis



Model Selection

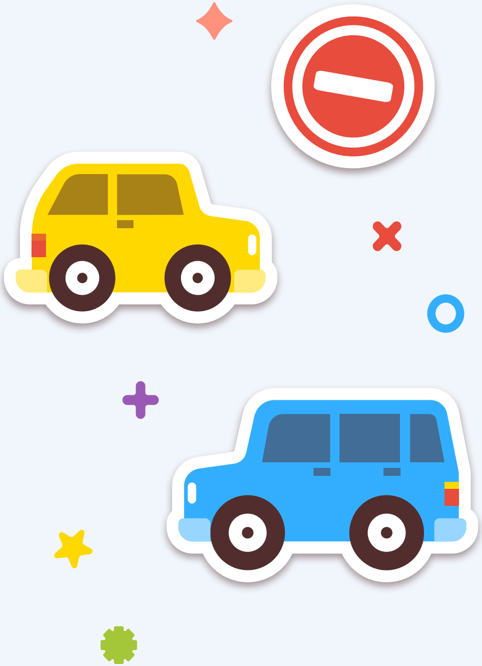
- All models had low accuracy, with some performing significantly worse (under 40% accuracy)
- RandomForest generally performed the worst, except when using OneRAttributeEval.
- OneRAttributeEval selection improved model performance
- KStar with OneRAttributeEval was the most accurate model with an accuracy of 47.54%, a FP rate of 0.420, ROC area of 0.620, and TP Rate of 0.459
- RandomForest with OneRAttributeEval had an accuracy of 45.9%, but it had a lower FP Rate (0.354), higher ROC area (0.631), and better TP Rate (0.475).
- **Chosen model: RandomForest with OneRAttributeEval Selection**
    - Pedestrian Location, Traffic Control, Surface Condition, Weather, Light, Agency Name, Safety Equipment, At Fault.

Evaluation

- Given the low performance, model not yet suitable for real-world applications
    - Model might not be capturing complex relationships between features, which could be crucial for improving predictive accuracy.
        - Lack of interaction between attributes in our model
    - Ambiguously adjacent labels with subtle differences in Injury Severity could have increased the likelihood of misclassification
        - E.g., labels such as “Possible Injury” vs. “Suspected Minor Injury” can make it hard for the model to draw clear boundaries
        - Difficult for models to accurately distinguish between them
        - Injury Severity can be subjective as two observers may label the same case differently
            - Added noise and inconsistency in the data.

08

Conclusion



What we learned

- For future iterations
  - Use Principal Component Analysis as our method for attribute selection, using linear combination of attributes to capture the relationships between attributes
  - Merge ambiguous adjacent labels to simplify the classification task
  - Explore ordinal classification methods, which recognize the inherent order among labels
    - More informed distinctions between injury severity, reducing confusion between adjacent labels and improving predictive accuracy.

Steps to recreate our Model

1. Open this folder with all of our data for this project: ML Q1 Project - Kaavya, Radin
2. Download crash_data.csv
3. Run the colab script to get train/test data, and download those files as train.csv and test.csf respectively
4. Open crash_data.csv in weka and in the “Select attributes” tab, select OneRAttributeEval and use Ranker as a search method
5. If “Injury Severity” is not already the class, then go to the Preprocess tab, click “Edit” and then right click on “Injury Severity” and select “Attribute as class.” Do this for all future files as well if Injury Severity is not already selected as the class
6. Run this, and select all attributes with a cutoff value of 44.5 or higher
7. open train.csv, and remove all attributes that were not selected by the OneRAttributeEval. save this as train1R.csv
8. Go to the “classify” tab, and choose Random Forest (under trees)
9. Select “Supplied test set”, choose test1R.csv, make sure “Injury Severity” is the class, and press start. If a warning comes up, press yes.

Thank you for listening!

Remember, don’t get hit by a car if you don’t want to be in a future dataset