Datamining Project: Chicago Crashes

Abdolrahim Tooranian[*]

March 20, 2025

1 Data Understanding and Feature Engineering

Most of the extracted features are not used due to various reasons, such as low variance or high cardinality, but they are kept in case they are needed later. These features are extracted from the *vehicles*, *people*, and *crashes* datasets. While many features were analyzed and ultimately excluded, here the focus is on the ones that were selected. Additionally, an HTML report is attached, which contains the distributions of the different features.

1.1 Missing and Erroneous Values

Several binary columns (e.g., `hit_and_run_i`) have a large number of missing values and very few "NO" entries. For instance, in `hit_and_run_i` (see Figure 1), more than 68% of the entries are missing, and "NO" makes up only about 1.3%. Accepting all missing values as valid would imply that over half of crashes in Chicago are hit-and-run, which is unlikely. Therefore, if these columns are used, the missing values are imputed as "NO," assuming a hit-and-run would be explicitly noted if it occurred.

Some columns contain out-of-range values (e.g., `latitude` and `longitude` in Figure 2 show locations in the ocean). There are also important features (e.g., `number_of_injuries` and `latitude`) with missing entries. Records with missing values in these crucial columns are removed.

1.2 Feature Engineering

Vehicles Dataset. From the vehicles data, `vehicle_year` is used to compute `vehicle_age`. The column `occupant_count` helps determine how many people are involved, and `towed_i` (possibly indicating severity) is processed as a binary feature (YES/NO), with missing values imputed as "NO."

[*]a.tooranian@studenti.unipi.it

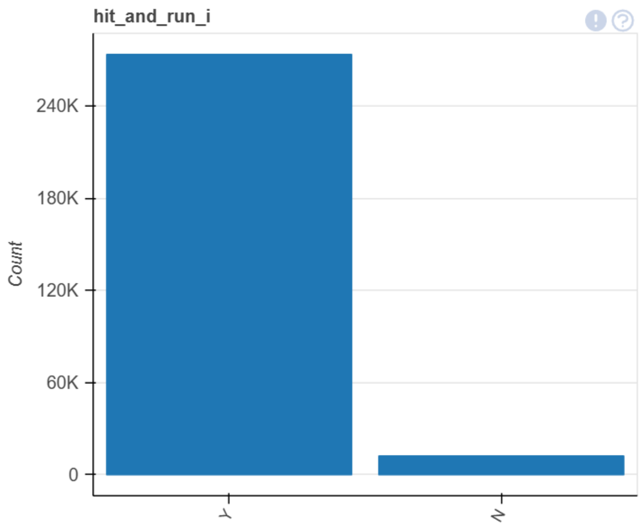


Figure 1: `hit_and_run_i` distribution.

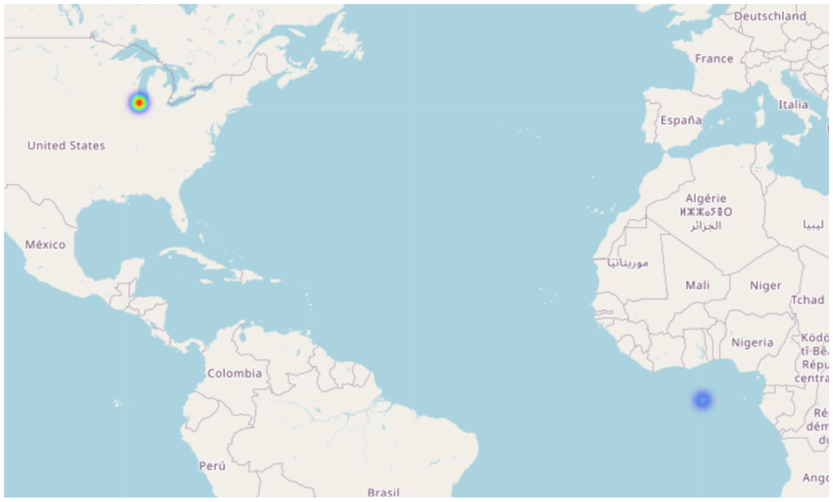


Figure 2: Out-of-range coordinates in the ocean.

People Dataset. Only drivers are retained, as other passengers have too many missing values. The columns `age` and `airbag_deployed` are used; all records with missing values (except for `age`) are removed. Missing `age` and `vehicle_age` are imputed using an iterative approach (`IterativeImputer`), leveraging data from the same vehicle or person. Afterwards, the *vehicles*, *people*, and *crashes* dataframes are merged for aggregation.

Categorical Variables. Many categorical columns have sparse categories. To avoid high dimensionality, the most frequent or semantically important category is identified, and a binary indicator is created. For example, `unit_type` (see Figure 3) is converted to `is_unit_driver`. Similar binary columns are created for other features (e.g., `vehicle_type`, `injury`, `airbag`, `ejection`, etc.).

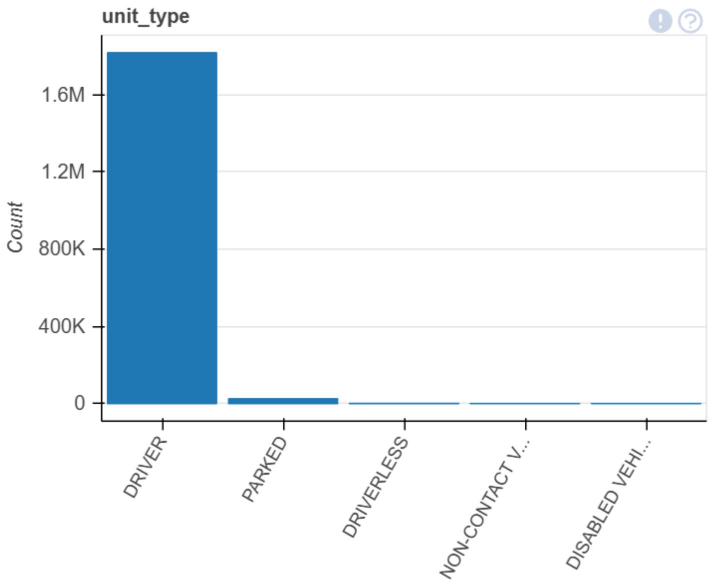


Figure 3: `unit_type` distribution.

1.3 Aggregation

At the crash level, the following features are created from the people and vehicle datasets:

- `total_occupant_count`: Sum of the occupants in all vehicles
- `avg_vehicle_year`: Average of `vehicle_year`
- `any_towed`: Indicates if any vehicle was towed
- `age_mean`: Mean age of drivers
- `airbags_deployed`: Number of airbags deployed

For the monthly profile, examples include:

- `total_crashes`
- `towed_percentage`

- injuries_total_percentage
- avg_total_occupant
- avg_num_units
- avg_car_age
- avg_speed_limit
- avg_airbag_deployed
- latitude, longitude
- damage_0, damage_1, damage_2: Breakdown of damage cost categories

1.4 Correlation Analysis

Correlation matrices are computed using Pearson and Spearman (see Figure 4). As expected, total_crashes strongly correlates with the damage columns, as they are essentially sums of the same events. Also, airbag_deployment is positively correlated with towing, which makes sense. Hence, towed is removed. Additionally, since car_age has some correlation with year, the latter is dropped.

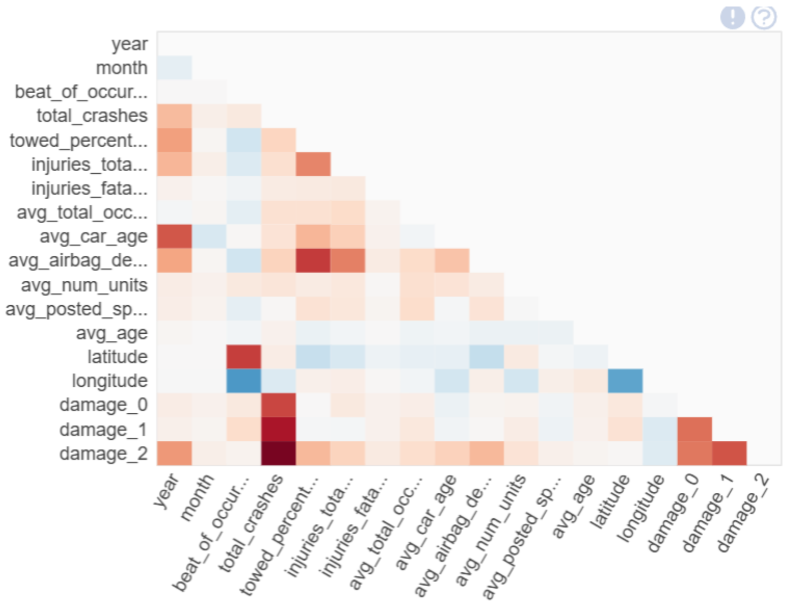


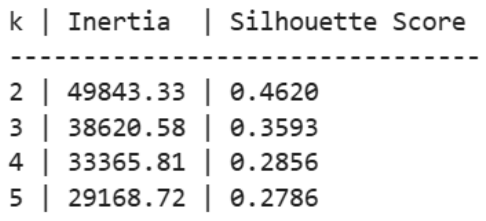
Figure 4: Correlation matrices (Pearson and Spearman).

2 Clustering

To address the problem with the damage columns, I initially explored using the mean of each damage category and then summing the results. However, this approach was problematic because the last category is an unbounded interval, and a mean value would not be a suitable representative.

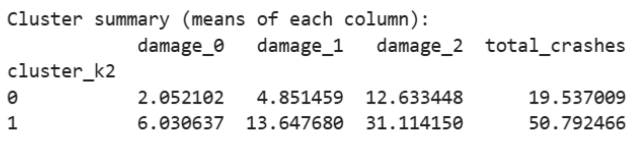
2.1 K-Means on Damage Columns

To avoid manual binning and also tackle the correlation between the damage columns, I applied k-means on the three damage variables. After checking silhouette (SIL) scores for small values of K, I found the highest score at $K = 2$ (Figure 5), which aligns perfectly with the goal of having two classes for subsequent classification. I then verified that these two clusters distinguish between higher and lower damage. Looking at the centroids (Figure 6), all damage values are noticeably lower in the first cluster. Additionally, the scatter plot (Figure 7) confirms that points with lower damage values form a single cluster. Consequently, I use this binary label to indicate high or low damage.



```
k | Inertia  | Silhouette Score
--------------------------------
2 | 49843.33 | 0.4620
3 | 38620.58 | 0.3593
4 | 33365.81 | 0.2856
5 | 29168.72 | 0.2786
```

Figure 5: Silhouette scores for different values of K; the highest is at $K = 2$.



```
Cluster summary (means of each column):
            damage_0   damage_1   damage_2  total_crashes
cluster_k2
0           2.052102   4.851459  12.633448      19.537009
1           6.030637  13.647680  31.114150      50.792466
```

Figure 6: Centroids for the two clusters after applying k-means on the damage columns.

2.2 Outlier Detection with DBSCAN

Next, I used DBSCAN with an ϵ value of 3.9, chosen by examining the distance plot (Figure 8). This resulted in identifying a small number of outliers.

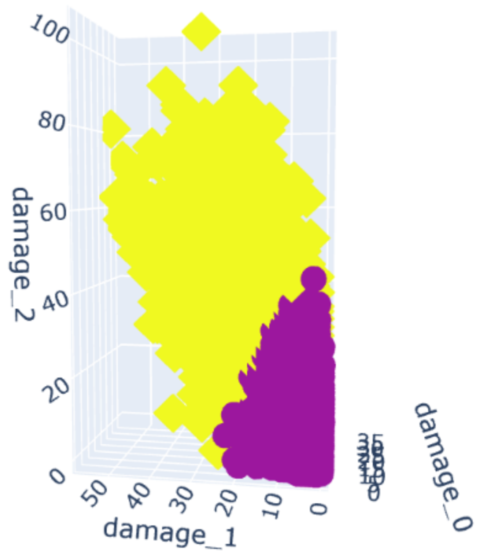


Figure 7: Scatter plot of the two damage clusters. Points with lower damage form one cluster.

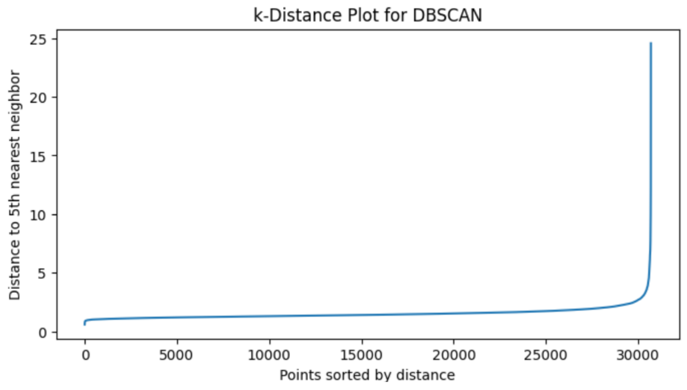


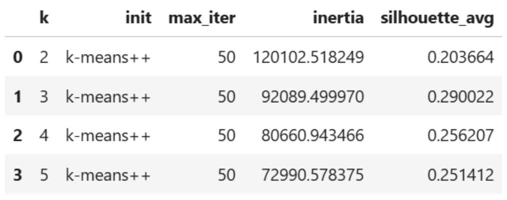
Figure 8: Distance plot used to select $\epsilon = 3.9$ for DBSCAN outlier detection.

2.3 K-Means on Additional Features

After experimenting with various features for k-means, I determined that the following columns have good potential for clustering:

`damage`, `avg_total_occupant`, `avg_car_age`, `avg_airbag_deployed`, `injuries_total_percentage`.

To choose K here, I examined metrics (Figure 9) showing that $K = 3$ provides a good balance, based on SIL scores and inertia. The elbow plot (Figure 10) also exhibits a steep drop at $K = 3$. The resulting centroids (Figure 11) suggest three clusters with increasing values for all columns, which can be interpreted as low, medium, and high "severity." Not always meaning higher cost as the centroid of the last cluster is almost 0.



	k	init	max_iter	inertia	silhouette_avg
0	2	k-means++	50	120102.518249	0.203664
1	3	k-means++	50	92089.499970	0.290022
2	4	k-means++	50	80660.943466	0.256207
3	5	k-means++	50	72990.578375	0.251412

Figure 9: Metrics table (SIL, inertia) indicating $K = 3$ is a good choice.

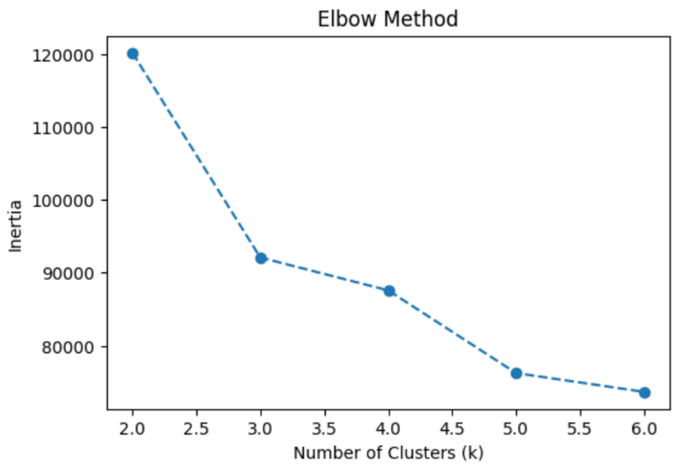
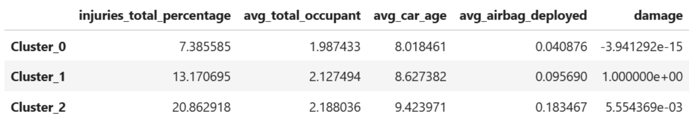


Figure 10: Elbow plot showing a notable drop in inertia at $K = 3$.



	injuries_total_percentage	avg_total_occupant	avg_car_age	avg_airbag_deployed	damage
Cluster_0	7.385585	1.987433	8.018461	0.040876	-3.941292e-15
Cluster_1	13.170695	2.127494	8.627382	0.095690	1.000000e+00
Cluster_2	20.862918	2.188036	9.423971	0.183467	5.554369e-03

Figure 11: Centroids of the three clusters on the selected features.

Because these clusters use features that also serve as input to downstream tasks, they will not be used directly for classification. To confirm this interpretation, we analyze the cluster distributions and visualize them. Figure 12 shows a 2D scatter plot of `avg_airbag_deployed` vs. `injuries_total_percentage` colored by cluster.

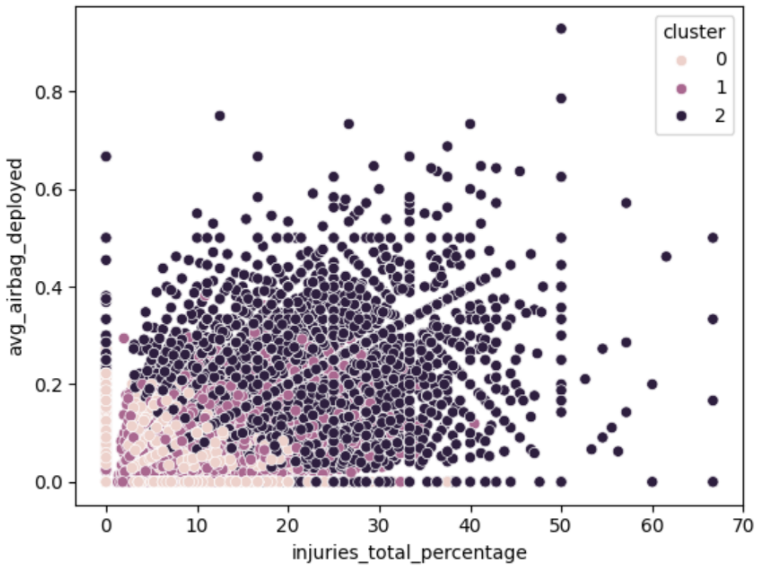


Figure 12: 2D scatter plot (airbag vs. injury) with points colored by cluster.

Figure 13 illustrates that `total_number_of_crashes` was not part of the clustering input but still shows interesting differences across clusters. The third cluster tends to have more severe crashes rather than a larger number of crashes. This implies that the "high-severity" group experiences intense accidents that may be very costly, even though their overall crash count is not necessarily high. However, since the original dataset only labeled crashes above $1,500 as "high damage," we do not have exact cost estimates for these very intense crashes. It is reasonable to suspect costs would be much higher than $1,500.

Additionally, Figure 14 presents a box plot for `avg_car_age` by cluster. Older vehicles seem more prevalent in the higher-severity clusters, though further analysis (e.g., linking neighborhood data or safety measures) would be needed to confirm why. We have this kind of linear relationship with the clusters for other features too.

2.4 DBSCAN on Extended Feature Set

I explored with different values of epsilon and increased it to, to solve the problem of highly imbalanced clusters as it was often the result of this algorithm. Re-examining the new distance plot (Figure 15) and increasing `min_samples` still generally led to clusters similar to the binary split on damage columns (Figure 16). Therefore, DBSCAN does not seem well-suited for these features, and more thorough analysis is necessary.

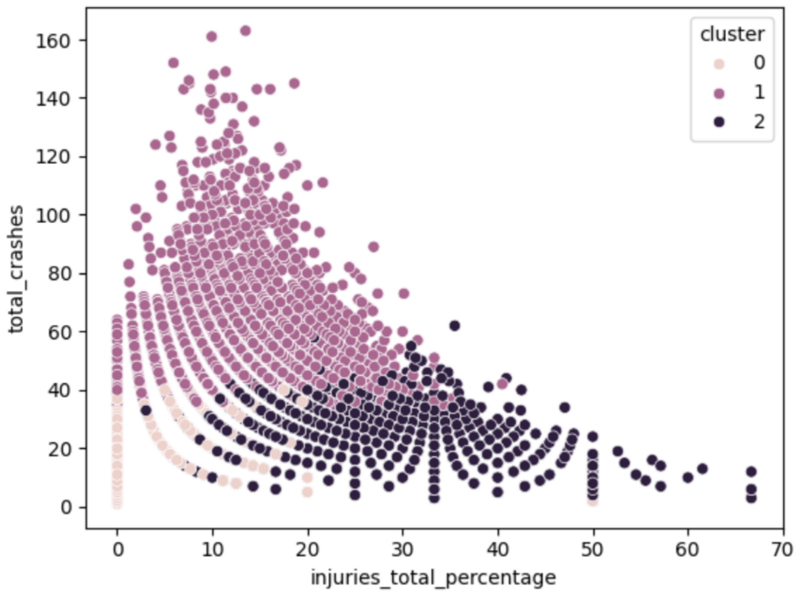


Figure 13: Total number of crashes (not used in clustering) and injuries across different clusters.

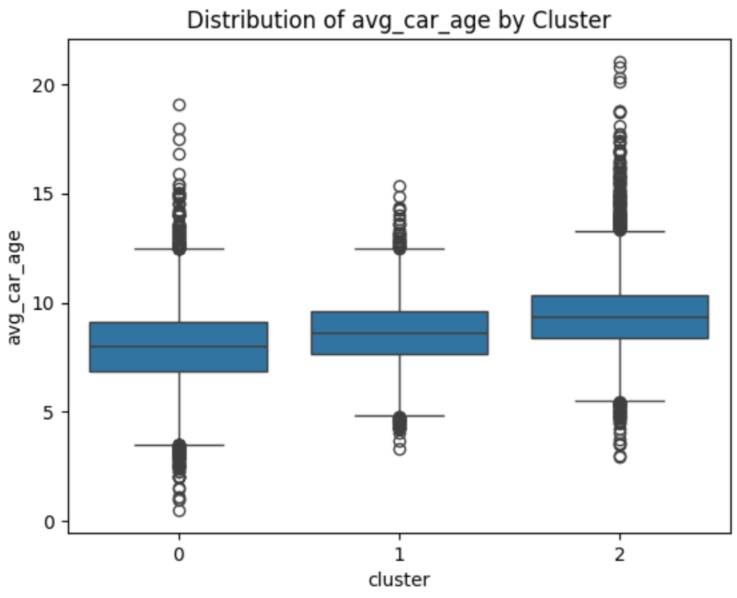


Figure 14: Box plot of average car age by cluster.

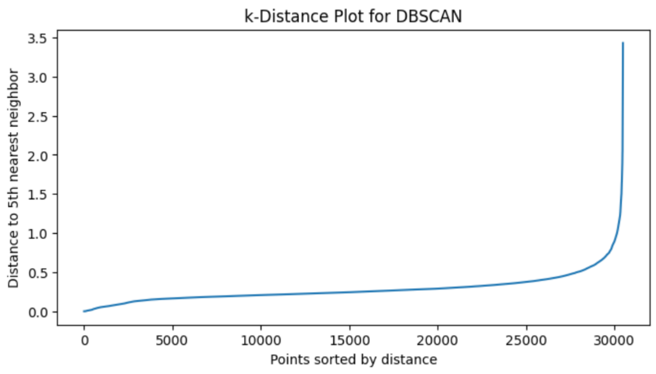


Figure 15: Sorted distance plot for DBSCAN on the extended feature set.

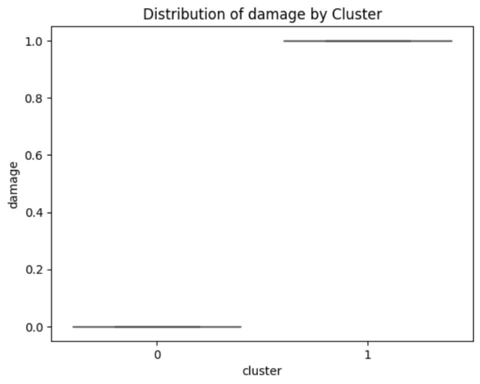


Figure 16: Box plot of `damage` for each cluster.

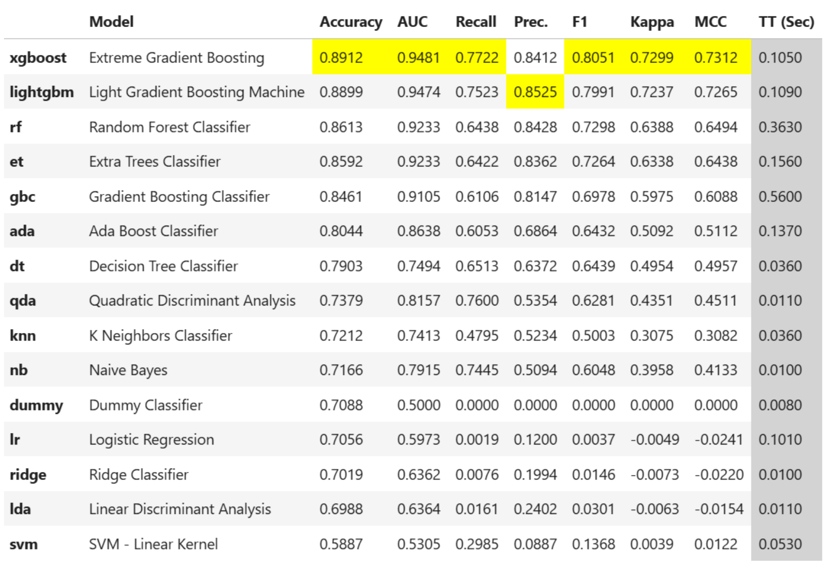
3 Classification

The aim is to classify the data into two classes that were discovered earlier via a simple k-means on damages (High vs Low). The features used to derive the damage classes and total number of crashes are excluded from this classification task, but the other features remain available.

The dataset is split into three subsets: training, validation, and test. I use the training set for hyperparameter tuning with k-fold cross-validation (CV), the validation set for model selection, and finally evaluate the chosen model on the test set.

3.1 Initial Model Exploration

I first ran a range of models with various hyperparameters (see Figure 17), using 6-fold CV to understand overall performance. Ensemble methods were the strongest. Some manual feature selection was also done to keep a reasonable subset of features. Since the performance metrics looked stable and there was no large gap between precision and recall, I did not spend time addressing class imbalance.



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.8912	0.9481	0.7722	0.8412	0.8051	0.7299	0.7312	0.1050
lightgbm	Light Gradient Boosting Machine	0.8899	0.9474	0.7523	0.8525	0.7991	0.7237	0.7265	0.1090
rf	Random Forest Classifier	0.8613	0.9233	0.6438	0.8428	0.7298	0.6388	0.6494	0.3630
et	Extra Trees Classifier	0.8592	0.9233	0.6422	0.8362	0.7264	0.6338	0.6438	0.1560
gbc	Gradient Boosting Classifier	0.8461	0.9105	0.6106	0.8147	0.6978	0.5975	0.6088	0.5600
ada	Ada Boost Classifier	0.8044	0.8638	0.6053	0.6864	0.6432	0.5092	0.5112	0.1370
dt	Decision Tree Classifier	0.7903	0.7494	0.6513	0.6372	0.6439	0.4954	0.4957	0.0360
qda	Quadratic Discriminant Analysis	0.7379	0.8157	0.7600	0.5354	0.6281	0.4351	0.4511	0.0110
knn	K Neighbors Classifier	0.7212	0.7413	0.4795	0.5234	0.5003	0.3075	0.3082	0.0360
nb	Naive Bayes	0.7166	0.7915	0.7445	0.5094	0.6048	0.3958	0.4133	0.0100
dummy	Dummy Classifier	0.7088	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0080
lr	Logistic Regression	0.7056	0.5973	0.0019	0.1200	0.0037	-0.0049	-0.0241	0.1010
ridge	Ridge Classifier	0.7019	0.6362	0.0076	0.1994	0.0146	-0.0073	-0.0220	0.0100
lda	Linear Discriminant Analysis	0.6988	0.6364	0.0161	0.2402	0.0301	-0.0063	-0.0154	0.0110
svm	SVM - Linear Kernel	0.5887	0.5305	0.2985	0.0887	0.1368	0.0039	0.0122	0.0530

Figure 17: Initial model comparison using 6-fold CV.

3.2 Hyperparameter Tuning

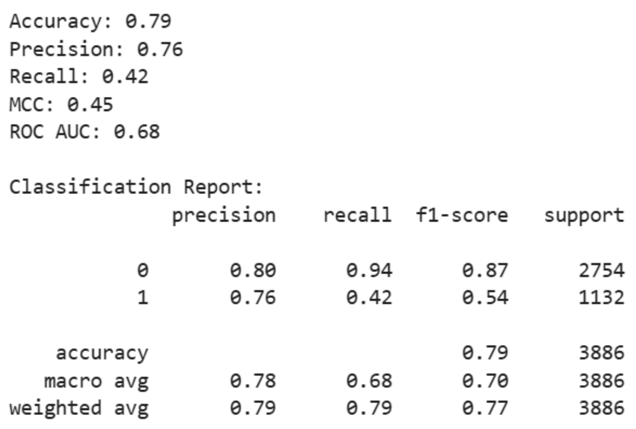
I then performed a more detailed tuning phase for KNN, Decision Tree, and an Ensemble method using Optuna, an open-source tool for hyperparameter optimization. I used the Matthews Correlation Coefficient (MCC) as the metric, since it takes into account both classes (unlike simple accuracy) and all four values in the confusion matrix.

3.2.1 Decision Tree

Various hyperparameters were tuned (details can be found in `task3.ipynb`). Figure 18 shows the validation results and Figure 19 shows the training scores for the best model. A visualization of the tree (Figure 20) shows it is not very large or deep.

3.2.2 KNN

For KNN, I tried different numbers of neighbors (1 to 50) and both Manhattan and Euclidean distances. Overall, the results were weaker than those of other models (see Figure 21).



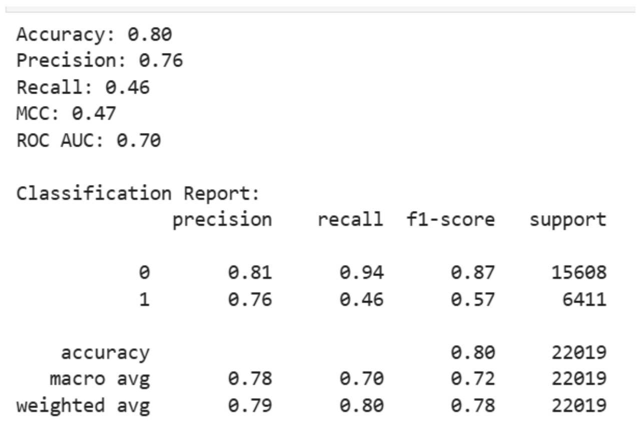
```
Accuracy: 0.79
Precision: 0.76
Recall: 0.42
MCC: 0.45
ROC AUC: 0.68

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.94      0.87      2754
           1       0.76      0.42      0.54      1132

    accuracy                           0.79      3886
   macro avg       0.78      0.68      0.70      3886
weighted avg       0.79      0.79      0.77      3886
```

Figure 18: Decision Tree validation results.



```
Accuracy: 0.80
Precision: 0.76
Recall: 0.46
MCC: 0.47
ROC AUC: 0.70

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.94      0.87     15608
           1       0.76      0.46      0.57      6411

    accuracy                           0.80     22019
   macro avg       0.78      0.70      0.72     22019
weighted avg       0.79      0.80      0.78     22019
```

Figure 19: Decision Tree training CV performance.

3.2.3 Ensemble Methods

I tested several ensemble methods. A Random Forest model reached an MCC of around 0.67 on the training set and slightly less on the validation set (Figure 22). XGBoost and LightGBM performed similarly, but LightGBM was faster to train. In the end, I chose XGBoost for slightly better interpretability and its tendency to avoid overfitting more than LightGBM.

Figure 23 shows the XGBoost final test performance. The confusion matrix is in Figure 24, and the ROC curve is in Figure 25. Also, the training performance is shown in Figure 26. The feature importance plot (Figure 27) indicates that features like airbag deployment and injuries are highly predictive of higher damage. Location is also important, as suggested by Figure 28, where high-damage crashes appear in certain regions.

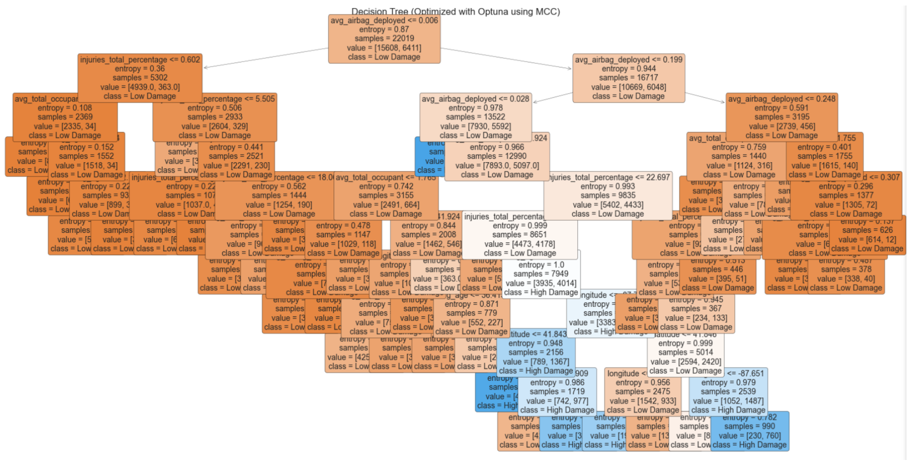
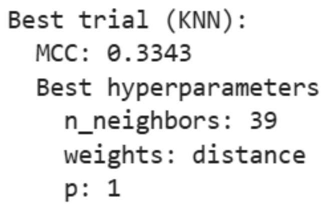
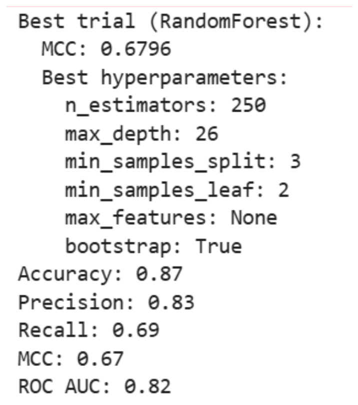


Figure 20: Decision Tree structure visualization.



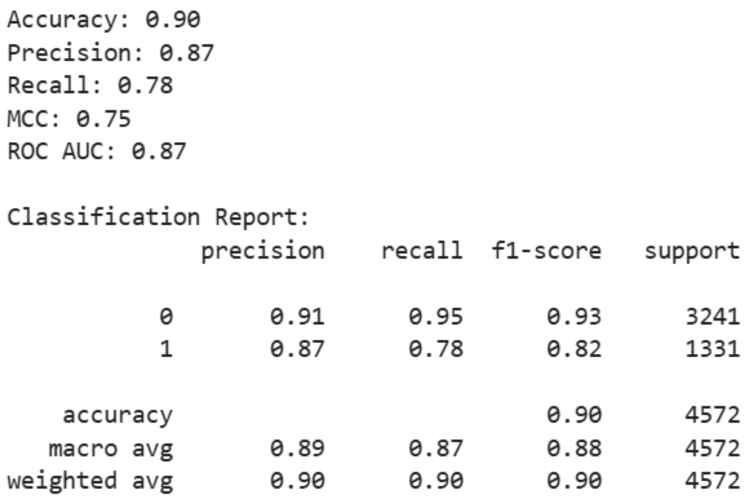
```
Best trial (KNN):
  MCC: 0.3343
  Best hyperparameters:
    n_neighbors: 39
    weights: distance
    p: 1
```

Figure 21: Best KNN model CV performance on the training set.



```
Best trial (RandomForest):
  MCC: 0.6796
  Best hyperparameters:
    n_estimators: 250
    max_depth: 26
    min_samples_split: 3
    min_samples_leaf: 2
    max_features: None
    bootstrap: True
Accuracy: 0.87
Precision: 0.83
Recall: 0.69
MCC: 0.67
ROC AUC: 0.82
```

Figure 22: Random Forest model performance on the validation set.



```
Accuracy: 0.90
Precision: 0.87
Recall: 0.78
MCC: 0.75
ROC AUC: 0.87

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      3241
           1       0.87      0.78      0.82      1331

    accuracy                           0.90      4572
   macro avg       0.89      0.87      0.88      4572
weighted avg       0.90      0.90      0.90      4572
```

Figure 23: XGBoost Final test results.

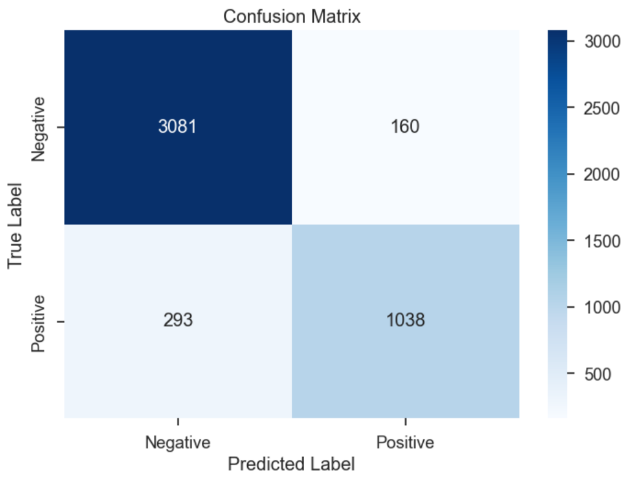


Figure 24: XGBoost confusion matrix on the test set.

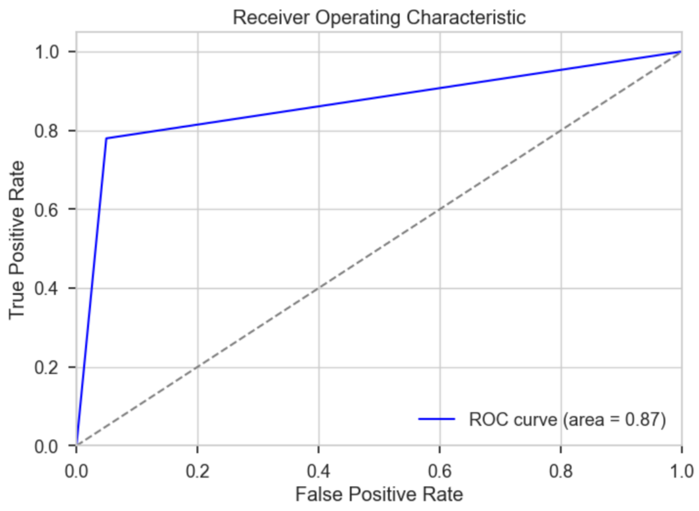
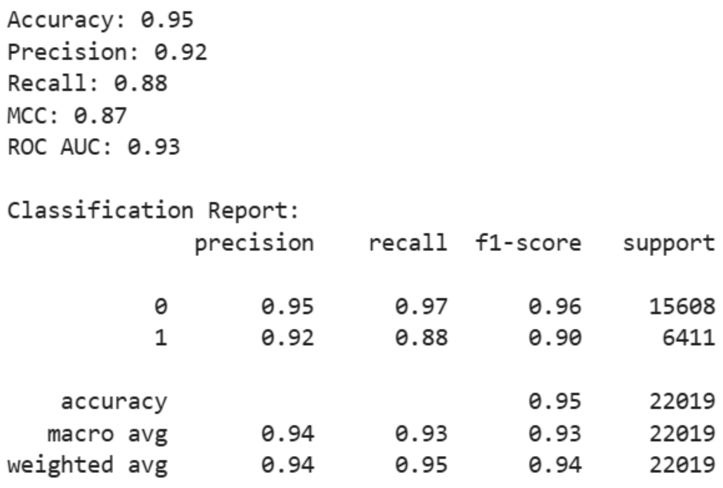


Figure 25: XGBoost ROC curve on the test set.



```
Accuracy: 0.95
Precision: 0.92
Recall: 0.88
MCC: 0.87
ROC AUC: 0.93

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     15608
           1       0.92      0.88      0.90      6411

    accuracy                           0.95     22019
   macro avg       0.94      0.93      0.93     22019
weighted avg       0.94      0.95      0.94     22019
```

Figure 26: XGBoost training set performance.

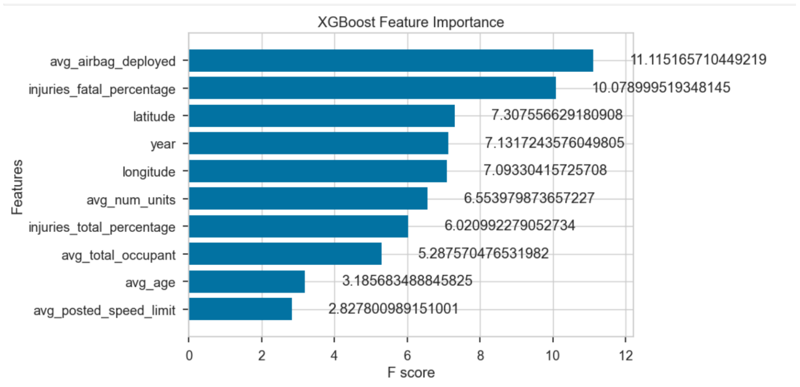


Figure 27: Feature importance of the final XGBoost model.

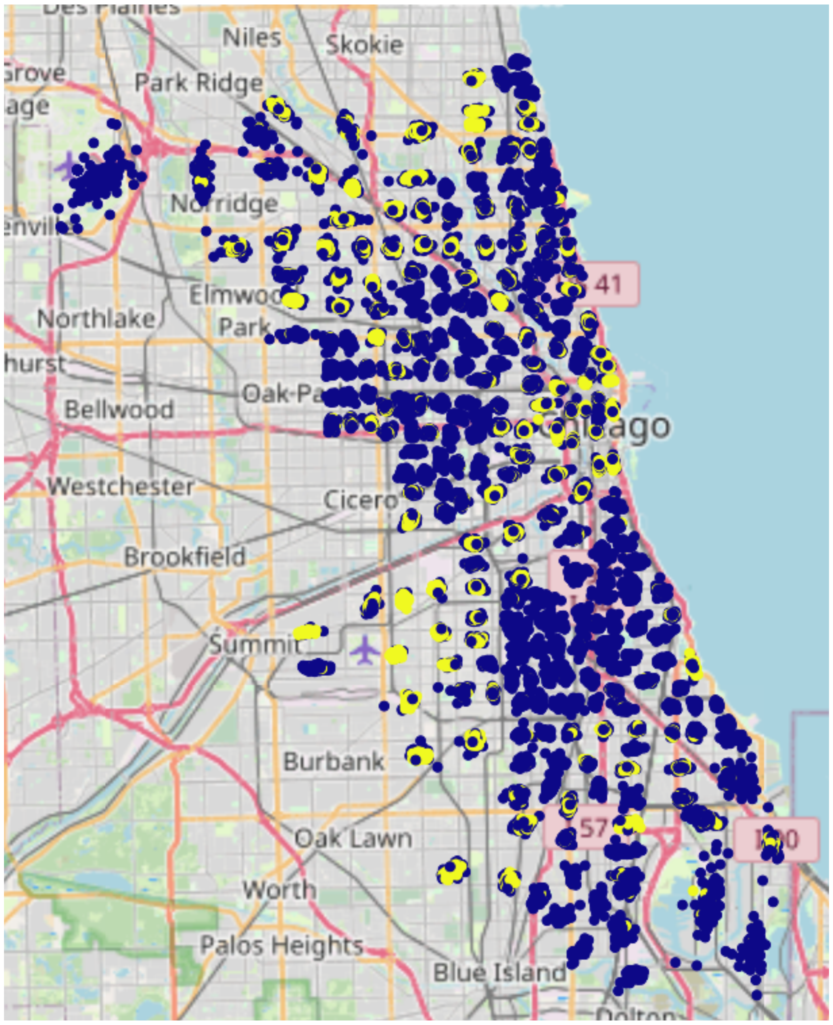


Figure 28: High-damage crashes (in yellow) on a city map.