Predicting Car Accident Severity

AI project for Hanyang ITE3051

Gavin Pryor, Department of Financial Management, gavin.anaiah@gmail.com
Randy Fu, Department of Computer Science, randyfu333@hanyang.ac.kr
Fatima Zahra El Bajdali,Department of Computer Science, bfz2005@hanyang.ac.kr
Marwa Errahmani Department of Computer Science, marwaerrahmani111@gmail.com

Our project intends to investigate the factors in a car accident and predict the severity of the accident using real world data and deep learning techniques. Specifically, we want to highlight which factors have the greatest impact on severity and have an AI model predict how severe an accident is based on those factors. This can later be integrated into a type of alert system, where alerts can be sent to would-be drivers warning them of the road conditions and how severe an accident could be if they get into one. We intend to train a neural network model capable of classifying accident severity into multiple levels (minor, moderate, or severe). We will use a dataset from Kaggle, which contains detailed records of traffic incidents across the United States. The dataset includes factors such as weather conditions, temperature, visibility, time of day, and road type all of which may influence accident outcomes.

Introduction

Road traffic accidents remain a major public-safety concern worldwide. Beyond the human cost, they create economic losses, traffic congestion, and significant pressure on emergency-response systems. Importantly, not all accidents have the same impact: some are minor incidents, while others result in serious injury or even fatalities. Being able to predict the likely severity of an accident using contextual information, such as weather, time of day, road type, visibility conditions, and local accident density, can support faster response prioritization, smarter warning systems, and more informed planning decisions for both drivers and infrastructure managers.

This project aims to develop and evaluate a machine-learning pipeline capable of predicting accident severity (minor, moderate, or severe) from real-world traffic incident data by learning patterns from environmental conditions, roadway characteristics, spatial-temporal context, and surrounding accident history. The system is designed not only to classify the severity level but also to understand how these diverse factors interact, allowing it to generalize to new situations and provide meaningful insights about the conditions under which severe accidents are most likely to occur. We also create neighborhood level descriptors such as kernel density estimates (KDE) and grid-cell accident statistics to capture the influence of local accident patterns. These features are then used to train deep learning classifiers alongside baseline models, allowing us to compare performance across architectures.

An additional objective of this project is to understand which factors contribute most to accident severity. After training, we conduct interpretability analyses, such as feature importance evaluations and model behavior visualizations, to identify the signals that most strongly shape the model’s predictions. Finally, the project includes a short demonstration video and a clear, structured technical blog that explain the model, methods, and findings, as well as a simple concept for how such a predictive system could be integrated into real-world notifications to drivers, predictive rerouting for GPS systems, or dynamic traffic-management dashboards for city planners.

Dataset

The dataset we are using for this project is the US Accidents (2016 - 2023) dataset found on Kaggle. This dataset contains around 7.7 million accident records across 49 US states from February 2016 to March 2023, using multiple APIs that provide streaming traffic incident data. These APIs broadcast traffic data captured by various entities, including the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road networks.

Our target feature and what we are predicting is Severity, which is a categorical variable with possible values of 1 (least severe), 2, 3, or 4 (most severe). In this case a "severe" accident is one that causes a longer traffic delay, with heavy congestion and long backups.

Following our processing the data, we use four broad groups of features to predict severity:

1. Time and location features that capture when and where the accident occurred. Examples include time of the accident, geographic coordinates of the crash location, and a spatial grid cells representing the area surrounding the crash.

2. Weather and environmental features, which describe the driving conditions at the time of the incident. Examples are:

- Temperature, humidity, and air pressure
- Visibility and wind conditions
- Rain, snow, or other forms of precipitation
- Indicators for clean weather, windy weather, thunderstorms, or weather intensity

3. Roadway and infrastructure features, representing characteristics of the road and nearby traffic elements, such as the presence of a crossing, junction, bump, or roundabout or, whether there is a stop sign or traffic signal

4. Spatial density and contextual features, summarizing patterns of nearby accidents and the severity of past incidents in the same area, such as local accident density within small spatial grids

These groups together give the model information about time, weather, road design, and spatial context that help it predict how severe each accident is likely to be. The original dataset, along with the full list of original features and their descriptions, can be found here.

Methodology

Data Processing

To process the data, we had to drop unnecessary columns, including leaky variables and redundant time or location fields, so we could focus on the timestamp and our feature-engineered spatial variables derived from latitude and longitude.

```
# drop unnessecary columns
cleaned = data.drop(columns={
    'Source',
    'End_Lat',
    'End_Lng',
    'End_Time',
    'Distance(mi)',
    'Description',
    'Street',
    'City',
    'Country',
    'County',
    'State',
    'Zipcode',
    'Country',
    'Timezone',
    'Airport_Code',
    'Weather_Timestamp',
    'Amenity',
    'Civil_Twilight',
    'Nautical_Twilight',
    'Astronomical_Twilight'
})
```

Using the raw lat/lng values, we projected coordinates into a metric CRS, created point geometries, generated multi-scale grid cells, and computed KDE-based density features to capture both local and regional accident patterns. We also handled missing values in weather and environmental columns, removed invalid or extreme outliers, and standardized continuous fields where needed. For the weather-related variables, we condensed dozens of noisy text categories into cleaner groups like weather type, intensity, and flags for windy or thunder conditions, which made the variables much easier for the model to learn from. Finally, since less severe accidents were far more common than severe accidents in the dataset, we utilized random undersampling to get an even number of each Severity class. This ensured that the model could not simply learn to favor the majority classes, which would have produced deceptively high accuracy while failing to recognize the rarer, high-severity cases. By training on a balanced dataset, each severity level contributed equally to the learning process, improving the model's ability to generalize and leading to a more meaningful and fair comparison across classes.

All of this gave us a cleaner, more meaningful version of the dataset that better reflects real accident conditions and sets the groundwork for stronger modeling.

Modeling

To predict Severity, we tested 3 different models and compared performance for each: **Random Forest**, **XGBoost**, and an **attention based neural network**. Our goal was to find the ideal balance between model performance and runtime using our laptops, while allowing us to view feature importance as an output.

To train our models, we utilized Sci-kit learn for random forest and xgboost, and tensorflow/keras for the neural network. Our train-test split was a stratified 70/30 split. Below is a brief description of our workflow for each of the three models:

Random Forest:

XGBoost:

Neural Network:

Evaluation & Analysis:

Notes

- Explanation of evaluation metrics and why we picked the ones we picked
- evaluation metrics of final model and real-world interpretation metrics (what do they tell us?)
- feature importance comparison of 3 initial models (maybe move the comparison ones up to other section)
- feature importance of final model
- runtime comparison of 3 initial models
- runtime of final model
- decision threshold discussion and real-world interpretation

Below is a summary of each of the models and their performance:

Model	Accuracy
XGBoost	0.71
Attention NN	0.6124
Random Forest	0.555

1. XGBoost Results:

XGBoost Model Accuracy: 0.7094

XGBoost Classification Report: precision recall f1-score support

```
       0       0.78      0.80      0.79     20209
       1       0.70      0.59      0.64     20209
       2       0.65      0.77      0.70     20209
       3       0.72      0.68      0.70     20208

accuracy                           0.71     80835
macro avg       0.71      0.71      0.71     80835
weighted avg       0.71      0.71      0.71     80835
```

XGBoost Confusion Matrix:
[[16116 1859 1411 823]
[2836 11830 3153 2390]
[879 1533 15598 2199]
[779 1699 3931 13799]]

2. Attention Neural Network Results:

Test macro F1: 0.7146090213946625

Classification report: precision recall f1-score support

```
       0       0.79      0.82      0.80     20209
       1       0.70      0.59      0.64     20209
       2       0.66      0.76      0.71     20209
       3       0.72      0.70      0.71     20208

accuracy                           0.72     80835
macro avg       0.72      0.72      0.71     80835
weighted avg       0.72      0.72      0.71     80835
```

Confusion matrix:
[[16475 1787 1221 726]
[2751 12000 3041 2417]
[897 1660 15354 2298]
[763 1748 3631 14066]]

3. Random Forest Results: Accuracy: 0.5508

Classification Report: precision recall f1-score support

```
       1       0.56      0.67      0.61     20209
       2       0.65      0.17      0.27     20209
       3       0.52      0.81      0.63     20209
       4       0.56      0.55      0.56     20208

accuracy                           0.55     80835
macro avg       0.57      0.55      0.52     80835
weighted avg       0.57      0.55      0.52     80835
```

Confusion Matrix:
[[13555 1014 2641 2999]
[6525 3425 6261 3998]
[1888 283 16460 1578]
[2108 569 6448 11083]]

Looking at the initial results of our three models, **XGBoost** and **Attention Based Neural Network** had similar macro-f1 scores at 0.71 while **Random Forest** had a macro-f1 score of 0.52. These results were not unexpected, as **XGBoost** excels at structued and tabular data compared to **Random Forest** and an **Attention Based Neural Network** is a very powerful architecture. However the **Attention Based Neural Network** took substantially longer to run compared to the **XGBoost** model. Given these results, we decided to focus on improving our XGBoost model's performance as described in the modeling section. To do so, we created a parameter sweep using Wandb for analysis to try and find the optimal parameters for our XGBoost model given our dataset. However, after running through the sweep the accuracy of the model did not dramatically increase, instead hovering around 0.72. The results of our sweeps are shown in the following graphs:

f1-weighted graph f1-macro graph raw f1-score graph

We evaluated off of macro-f1 score because of the nature of our data. We chose to evaluate our model using the f1 score because it provides a balanced measure of precision and recall. Unlike accuracy, which can be misleading in cases of uneven error types, the F1 score ensures that the model performs well both in correctly identifying positive cases (recall) and in minimizing false positives (precision). This makes it a more robust metric for assessing overall classification performance, especially when both types of errors carry meaningful consequences.

The parameter importance for f1-macro is show below. The most important parameters are listed at the top, and the correlation is listed as either positive(green) or negative(red) respectively:

f1-macro parameter importance

Combining a GridSearch and sweep, we ended up with the final model results:
Best params: {'xgb__gamma': 0.5, 'xgb__max_depth': 7, 'xgb__subsample': 0.7}
Best CV macro F1: 0.7132117427623111
Test macro F1: 0.7146090213946625

Classification report: precision recall f1-score support

```
          0       0.79      0.82      0.80     20209
          1       0.70      0.59      0.64     20209
          2       0.66      0.76      0.71     20209
          3       0.72      0.70      0.71     20208

   accuracy                           0.72     80835
   macro avg       0.72      0.72      0.71     80835
   weighted avg       0.72      0.72      0.71     80835
```

Confusion matrix:
[[16475 1787 1221 726]
[2751 12000 3041 2417]
[897 1660 15354 2298]
[763 1748 3631 14066]]

Even with parameter sweeps and cross-validation to find the best parameters, we saw little to no improvement in our accuracy or f1 score. These results indicate to us that our current features may not carry enough predictive power to increase our prediction accuracy. More feature engineering or more data may be required to improve the performance of our model.

Related Work

We referenced the following sources:

Datasets & Documentation:

- US Accidents (2016–2023) — Kaggle

Methods & Technical Resources:

- XGBoost official documentation
- Scikit-learn documentation
- TensorFlow/Keras guides for custom layers
- Blog posts/tutorials on: Attention mechanisms for tabular data

Tools Utilized

- Python 3.10
- Pandas, NumPy, Matplotlib, Seaborn
- scikit-learn
- TensorFlow / Keras
- XGBoost
- Weights & Biases
- PyTorch

Conclusion

Notes

- summary and wrap up
- real world impact and applications
- deployment feasability

Through this project, we were able to explore how different environmental, temporal, and spatial factors contribute to the severity of car accidents. By testing multiple machine learning models, we gained a better understanding of what types of algorithms work best for this kind of structured, real world data.

Overall, XGBoost performed the best, reaching about 71% accuracy, which makes sense because boosted tree models are known to handle tabular and mixed type features very well. Our attention based neural network did not reach the same level of accuracy, but it was still useful because it helped us interpret which features the model considered important. The Random Forest served as a good baseline but showed lower performance compared to the other two.

From the results, we found that weather conditions, visibility, and local accident density played a big role in predicting severity. Features related to road infrastructure and time of day also contributed, but to a smaller degree. One of the challenges we faced was the moderate severity class, which was harder to predict correctly due to overlapping patterns with the other classes.