# Predictive Analytics in Road Safety

A Machine Learning Approach to Traffic Accident Analysis


Manan Jain
**University of Illinois Chicago**
*College of Engineering*
mjain35@uic.edu
657477565

Hemanth Nagulapalli
**University of Illinois Chicago**
*College of Engineering*
hnagul2@uic.edu
670292806

Sravan Nekkanti
**University of Illinois Chicago**
*College of Engineering*
dnekka2@uic.edu
668012713

Surya Reddy Nallamilli
**University of Illinois Chicago**
*College of Engineering*
snalla20@uic.edu
656163373


Project Folder Link

## I. Abstract


Road travel is the most common and most used means of travel. Therefore, road safety is of utmost importance. Every year, millions of accidents destroy both life and property. This project aims to study the different features and factors that go into road accidents and help predict the severity of accidents that certain circumstances may cause. Mitigating traffic accidents is a critical challenge for public safety, prompting extensive research into accident analysis and prediction over the last few decades. Shortcomings of existing studies include reliance on small-scale datasets with limited coverage, dependency on comprehensive data, and inapplicability for real-time purposes. In response to these challenges, this project introduces an innovative real-time traffic accident prediction solution using easily obtainable yet sparse data. Our approach centers around the Kaggle dataset US-Accidents [1], containing 7.7 million data records (2.9 GB). This data has been meticulously gathered from multiple sources, leading to an exhaustive study of road accidents. We will utilize machine learning techniques along with this data and try to predict the severity of accidents.


## II. CCS Concepts:

Machine Learning: - Code: I.2.6 (Learning)

Big Data Mining: - Code: H.2.8 (Database Applications - Data Mining)

Neural Networks: - Code: I.2.10 (Artificial Intelligence - Vision and Scene Understanding)

Theory of Computation: - Code: F.1 (Computation by Abstract Devices)

Data Integration: - Code: H.2.4 (Systems - Query processing)

Computing Methodologies: - Code: H.3 (Information Systems - Information Storage and Retrieval)

Supervised Learning by Classification: - Code: I.2.6 (Learning) and I.5.2 (Pattern Recognition)

Applied Computing: - Code: K (Computing Milieux)

Transportation: - Code: J.7 (Computers in Other Systems - Transportation)

## III. Introduction

Road travel is the predominant and widely adopted mode of transportation, underscoring the paramount importance of road safety. Annually, millions of accidents take a toll on human lives and property. This project explores the diverse features and factors influencing road accidents, aiming to forecast the severity of accidents arising from specific circumstances. Addressing the critical challenge of mitigating traffic accidents for public safety has spurred extensive research in accident analysis and prediction over recent decades. Existing studies have limitations such as reliance on small-scale datasets with restricted coverage, dependence on exhaustive data, and impracticality for real-time applications. In response, this project introduces an innovative solution for real-time traffic accident prediction using easily obtainable yet sparse data. Our methodology revolves around leveraging the Kaggle dataset US-Accidents[1], encompassing 7.7 million data records (2.9 GB). This dataset, meticulously compiled from multiple sources, facilitates a comprehensive exploration of road accidents. Employing various machine learning techniques alongside this data, we aim to predict the severity of imminent accidents.

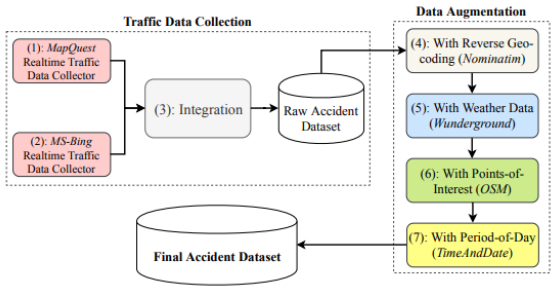
## IV. Literature Review

Addressing the global challenge of reducing traffic accidents, which claimed 1.25 million lives in 2013 alone, remains a crucial public safety concern. The escalating number of deaths in 68 countries between 2010 and 2013 underscores the urgency for effective accident prediction. Accurate prediction is essential for optimizing public transportation, enhancing route safety, and cost-effectively improving transportation infrastructure. Consequently, extensive research has focused on analyzing the impact of environmental stimuli, predicting accident frequencies, and assessing the risk of accidents based on factors such as road network properties, weather conditions, and traffic patterns. To overcome the limitations of existing studies, a novel solution utilizing an exhaustive dataset and a machine learning approach incorporates various data categories such as traffic events, weather conditions, points of interest, and time. This model aims to provide real-time predictions for a defined geographical region during fine-grained time intervals. The subsequent sections of the paper cover preliminaries, related work, dataset construction, the accident prediction framework, experiments, and results, concluding with a summary of findings.

## V. Methodology

### V.i. Dataset[1]

The following diagram represents the data collected at US-Accidents[1].




We have utilized this dataset and provided our augmentation to it. The initial dataset comprises 7.7 million records, amounting to 2.9 GB. This dataset was extracted using Bing API and MapQuest. These sources provide the backbone of the Accident dataset. Further augmentation in the dataset comes from extracting information from various other sources, such as weather APIs, map metadata, etc. This concludes the dataset. The following table shows the features presented as such:

| # | Column | Data Type |
|---|---|---|
| 1 | ID | Object |
| 2 | Source | Object |
| 3 | Severity | Int64 |
| 4 | Start_Time | Object |
| 5 | End_time | Object |
| 6 | Start_Lat | Float64 |
| 7 | Start_Lng | Float64 |
| 8 | Distance(mi) | Float64 |
| 9 | Description | Object |
| 10 | Street | Object |
| 11 | City | Object |
| 12 | County | Object |
| 13 | State | Object |
| 14 | Zipcode | Object |
| 15 | Country | Object |
| 16 | TimeZone | Object |
| 17 | Airport_Code | Object |
| 18 | Weather_Timestamp | Object |
| 19 | Temperature(F) | Float64 |
| 20 | Wind_Chill(f) | Float64 |
| 21 | Humidity(%) | Float64 |
| 22 | Pressure(in) | Float64 |
| 23 | Visibility(mi) | Float64 |
| 24 | Wind_Direction | Object |
| 25 | Wind_Speed(mph) | Float64 |
| 26 | Precipitation(in) | Float64 |
| 27 | Weather_Condition | Object |
| 28 | Amenity | Bool |
| 29 | Bump | Bool |
| 30 | Crossing | Bool |
| 31 | Give_Way | Bool |
| 32 | Junction | Bool |
| 33 | No_Exit | Bool |
| 34 | Railway | Bool |
| 35 | Roundabout | Bool |
| 36 | Station | Bool |
| 37 | Stop | Bool |
| 38 | Traffic_Calming | Bool |
| 39 | Traffic_Signal | Bool |
| 40 | Turning_Loop | Bool |
| 41 | Sunrise_Sunset | Object |

Using all these features, we broadly classify them into three significant features, namely:

- Location: This comprises the GPS information along with the timestamps

- Weather: All the features of weather are grouped in this broad classification
- Road Conditions: The rest of the metadata are clubbed together here
- Severity: We are trying to predict this using our project. Severity is a metric that defines the overall losses sustained by that accident. Thus, it reflects the loss of life and property and the damages to the traffic like roadblocks, etc.

## V.ii. Preprocessing

The initial dataset presents us with two challenges:

- Since we have data from many different sources, we need to efficiently utilize the data and train models so that our machines can handle it with limited resources.
- Although it contains many features, there are many missing values.

We decided to approach the second problem. Since most of the missing values were in some form or the other in the weather-related features, we decided to use some available APIs to fill in those missing values. However, we saw that all the freely available APIs have a limit of about 1000 calls daily[3]. This severely limited the usefulness of this method since the missing values were around 1% of the entire dataset (about 70000). Not all these missing values were also present alongside each other. This meant that we would need 600000 calls to fill these up.

We devised an innovative approach to address missing values in our dataset. Leveraging the available GPS coordinates of each data point and applying the haversine formula, we identified all data points within a 5-mile radius of those with missing entries. Subsequently, utilizing timestamps, we computed three days centered around the data point with lost entries and derived averages for the corresponding values. This method effectively enabled us to impute missing data points with meaningful and contextually relevant information.

The *haversine formula* allows the haversine of $\theta$ (that is, $\operatorname{hav}(\theta)$) to be computed directly from the latitude (represented by $\varphi$) and longitude (represented by $\lambda$) of the two points:

$$\operatorname{hav}(\theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

where

- $\varphi_1$, $\varphi_2$ are the latitude of point 1 and latitude of point 2,
- $\lambda_1$, $\lambda_2$ are the longitude of point 1 and longitude of point 2.

To combat the first challenge of utilizing the entire dataset, we used a stratified sampling of 10% of the data and multiple passes to develop a valuable and accurate solution. Also, some columns not mentioned here, such as End_lat, End_lng, and Astronomical_Twilight, were dropped, as we believe these bring in less value overall compared to the other features. Furthermore, these features had a lot of missing entries.

Some further preprocessing was done to understand the data better and analyze the patterns presented. We grouped the data according to various cities as well as states. This analysis will be shown in the next section.

## V.iii. Models

We used multiple machine learning models to compare their efficiency and accuracy on this dataset. Each model was picked because we wanted a predictive analysis of the accidents. Since we have decided to take “Severity” as the metric, we want the models to classify the data points into the different “Severity” measures.

### Logistic Regression[4][5]

Logistic Regression is suitable for binary classification tasks, making it appropriate for predicting the severity of accidents (assuming the severity is binary, such as low or high). It is advantageous when there is a need for simplicity and interpretability in the model. Logistic Regression works well when the relationship between the input features and the target variable is approximately linear, and it can provide insights into the importance of different features in determining accident severity.

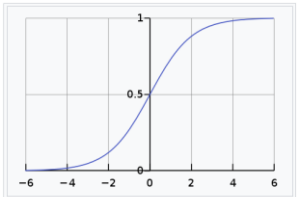
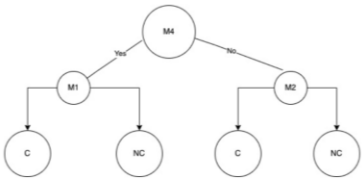


Figure 1. The standard logistic function $\sigma(t)$; $\sigma(t) \in (0,1)$ for all $t$.

However, since the classification here is not binary and the data is not necessarily linearly related, we assume this model would be the baseline or worst-performing model.

## Decision Trees[6][7]

Decision Trees are decisive for understanding decision-making processes. Decision Trees can be employed in this dataset to identify critical factors influencing accident severity. The visual representation of decision trees can provide insights into the hierarchy of features that contribute to the classification, aiding in the interpretability of the model.




## Random Forest[8][9]

Random Forest is a versatile ensemble learning technique for classification and regression tasks. It is well-suited for this dataset due to its ability to capture complex relationships and patterns. Random Forest can take a mix of numerical and categorical features, making it robust for datasets with diverse types of information, such as accident-related variables, weather conditions, and geographical coordinates.

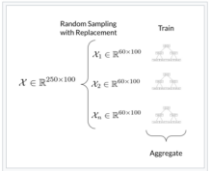
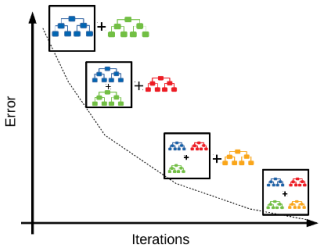



Illustration of training a Random Forest model. The training dataset (in this case, of 250 rows and 100 columns) is randomly sampled with replacement *n* times. Then, a decision tree is trained on each sample. Finally, for prediction, the results of all *n* trees are aggregated to produce a final decision.

Thus, we would assume this model would be one of the best-performing models.

## Gradient Boosting[10][11]

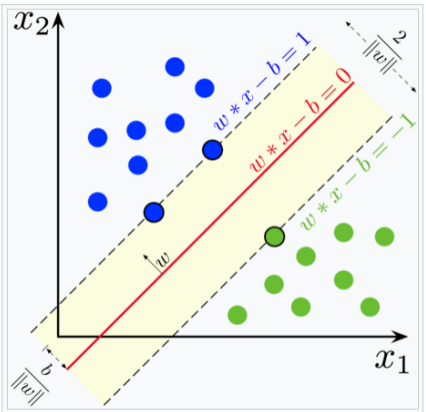
Gradient Boosting is beneficial when seeking to improve predictive accuracy by combining the strengths of multiple weak learners. Since the dataset predicts accident severity with diverse features, Gradient Boosting can enhance the model's ability to capture intricate patterns and dependencies within the data. It is beneficial for situations where subtle relationships between variables might be crucial.




Again, we would assume this model to be high-performing.

## Support Vector Machines[12][13]

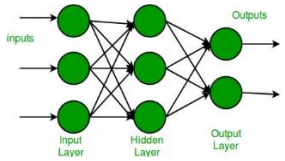
SVMs are effective for classification and regression tasks and work well with complex relationships and non-linear patterns. Given the diverse nature of your dataset, SVMs can be employed to separate different severity classes by finding an optimal hyperplane. SVMs are particularly useful when dealing with high-dimensional data and can effectively handle numerical and categorical features.




Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

## Multi-Layer Perceptron[14][15]

As an artificial neural network, Multi-Layer Perceptron is adept at capturing complex relationships and patterns within data. In this dataset, which includes various features like geographical coordinates, weather conditions, and binary indicators, an MLP can be beneficial for predicting accident severity by learning intricate dependencies among these diverse variables. However, MLPs may require careful tuning due to their complexity, and they shine when dealing with large, high-dimensional datasets.



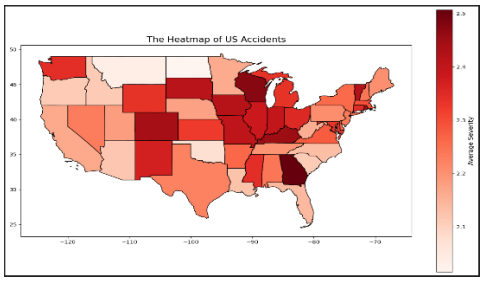

Since we would like to make a better prediction, we added this as a final model to our analysis. The only issue is that this is a weighty and challenging training model with limited resources and our lack of prior knowledge.

## VI. Data Analysis

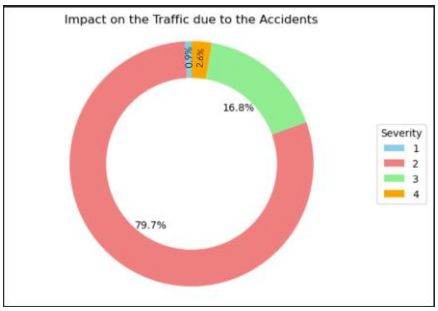
### VI.i. Visualisations[16][17]

First, we would like to present some visualizations of the dataset. There can be much more information and further visualizations; however, we only use a small number to keep it brief.

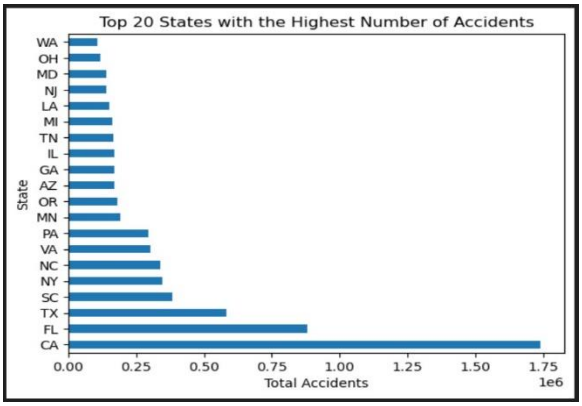
The following image shows the heatmap of the US based on the entire dataset. The map is generated by calculating the average "Severity" of accidents in each state of the USA and then plotting it on the map accordingly. The higher the average "Severity," the darker the map.



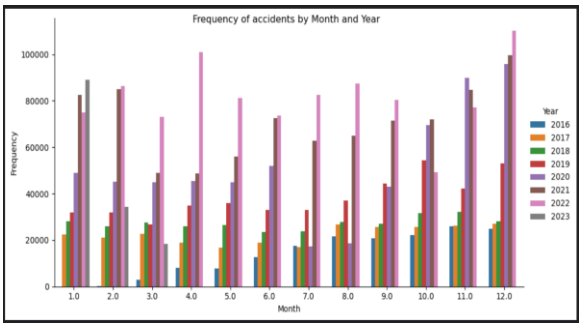

The following image shows the distribution of the accident data according to "Severity." This was done by grouping the entire dataset by "Severity" and plotting the counts.




The following image shows the top 20 accident-prone cities in the USA. This was plotted using the count of accidents in each town.




The following image shows the accidents spread over the years and months. It helps us understand the drastic increase in accidents over the years and highlights the need for more research into making the roads safer.




### VI.ii. Models and Tuning

After performing the preprocessing and getting all the data, we used the models suggested and tried to fit the data. We used a 30/70 data split for testing and training, respectively. As mentioned, since the data proved too big for our machines, we performed a stratified sampling to get 10% of it at a time and then fit the models. We used 5-fold cross-validation[19] to initialize the tuning of hyperparameters and then further utilized GridSearch[18] to optimize.

## VII. Results

### Random Forest

Accuracy: 91.88%

Precision: 0.92

Recall: 0.92
F1 Score: 0.92
Random Forest Cross-Validation Scores: 0.92 (mean) +/- 0.00 (std)

### Decision Tree

Accuracy: 89.85%
Precision: 0.90
Recall: 0.90
F1 Score: 0.90
Decision Tree Cross-Validation Scores: 0.89 (mean) +/- 0.00 (std)

### SVM

Accuracy: 84.65%
Precision: 0.83
Recall: 0.85
F1 Score: 0.84
SVM Cross-Validation Scores: 0.85 (mean) +/- 0.00 (std)

### Logistic Regression

Accuracy: 80.27%
Precision: 0.78
Recall: 0.80
F1 Score: 0.78
Logistic Regression Cross-Validation Scores: 0.81 (mean) +/- 0.00 (std)

### Gradient Boosting

Accuracy: 91.84%
Precision: 0.92
Recall: 0.92
F1 Score: 0.92
Gradient Boosting Cross-Validation Scores: 0.92 (mean) +/- 0.00 (std)

### MLP

Accuracy: 85.53%
Precision: 0.85
Recall: 0.86
F1 Score: 0.85
MLP Cross-Validation Scores: 0.89 (mean) +/- 0.00 (std)

## VII.i. After GridSearch Optimization

### Random Forest

Accuracy: 91.96%
Precision: 0.92
Recall: 0.92
F1 Score: 0.92
Random Forest GridSearch Scores: 0.92 (mean) +/- 0.00 (std)

### Decision Tree

Accuracy: 91.42%
Precision: 0.91
Recall: 0.91
F1 Score: 0.91
Decision Tree GridSearch Scores: 0.91 (mean) +/- 0.00 (std)

### SVM

Because of the nature of SVM's fit algorithm being $O(n^2)$, GridSearch could not be completed.

### Logistic Regression

Accuracy: 80.37%
Precision: 0.78
Recall: 0.80
F1 Score: 0.79
Logistic Regression GridSearch Scores: 0.81 (mean) +/- 0.00 (std)

### Gradient Boosting

Accuracy: 93.45%
Precision: 0.93
Recall: 0.93
F1 Score: 0.93
Gradient Boosting GridSearch Scores: 0.94 (mean) +/- 0.00 (std)

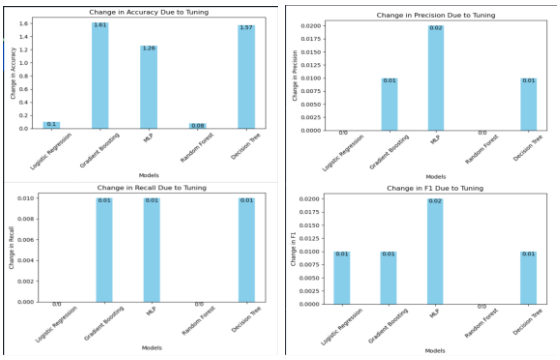
### MLP

Accuracy: 86.79%
Precision: 0.87
Recall: 0.87
F1 Score: 0.87
MLP GridSearch Scores: 0.89 (mean) +/- 0.00 (std)

## VIII. Inferences

Based on the results mentioned above, we can draw several conclusions.

First, a simple 5-fold CV showed us that Random forest was the best-performing model, followed by Gradient boosting. However, when we perform GridSearch, Gradient boosting becomes the best-performing model. The following images show the results of the hyperparameter tuning:




Another inference we can draw is that because of the overall size of the dataset, the models have all been trained above expectations. We reached 93.45% accuracy with gradient boosting in predicting the severity of the accidents.

Also, as expected, logistic regression, our baseline model, performed the worst. It showed only 80.37% accuracy in the predictions.

Furthermore, the MLP model that we trained did not perform as expected. This can be due to several reasons. However, we believe it may be due to proper tuning and parameter settings.

The models put the highest weights on the city and state regarding geographical features. The weather, visibility, and temperature features have been assigned the highest weights.

Unfortunately, the weights assigned to the metadata were less than we had assumed. This showed us that we had not adequately fine-tuned the model or missed some opportunity to account for the metadata in another way.

## IX. Conclusion and Future Work

In conclusion, we achieved an excellent overall performance, which was better than we had predicted. This could be because of the thoroughness of the dataset and because we could impute many of the missing data values.

However, there is a lot of scope for improvement in this project. For example, using cloud technologies could alleviate some of our resource issues. We can also utilize some paid APIs to fill in the missing values to make the predictions stronger. A further study that uses even sparser data is also possible. This can be helpful since acquiring data in real-time is not easy, and predictive analysis should be more robust and may be utilized in self-driving cars to mitigate road travel's dangers further.

## X. Acknowledgments

We thank Professor Lu Cheng for allowing us to work on this group project. This project helped us learn a lot about machine learning, big data mining concepts, and the importance of road safety.

## XI. References

[1]. https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents
[2]. https://en.wikipedia.org/wiki/Haversine_formula
[3]. https://openweathermap.org/api
[4]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[5]. https://en.wikipedia.org/wiki/Logistic_regression
[6]. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[7]. https://en.wikipedia.org/wiki/Decision_tree
[8]. https://en.wikipedia.org/wiki/Random_forest

[9]. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[10]. https://en.wikipedia.org/wiki/Gradient_boosting
[11]. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[12]. https://en.wikipedia.org/wiki/Support_vector_machine
[13]. https://scikit-learn.org/stable/modules/svm.html
[14]. https://en.wikipedia.org/wiki/Multilayer_perceptron
[15]. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[16]. https://pypi.org/project/folium/
[17]. https://matplotlib.org/stable/api/index
[18]. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[19]. https://scikit-learn.org/stable/modules/cross_validation.html
[20]. https://chat.openai.com/
[21]. https://app.grammarly.com/