Accident Severity Prediction through Machine Learning

Ismail Faiz
UC San Diego
ifaiz@ucsd.edu

Arnav Talreja
UC San Diego
atalreja@ucsd.edu

Abstract

We chose this project due to our passion for driving, cars, and motorsports, and our desire to contribute to road safety. Despite a decrease in fatal accidents since the start of the century, the number remains significant. Advances in vehicle design and engineering have played a key role in this improvement, but there's still more to be done. Leveraging a comprehensive dataset of 7.7 million accident records across the US from 2016 to 2023, we aim to develop a robust machine learning model to predict accident severity. This project utilizes the extensive resources at SDSC, enabling us to create meaningful and useful predictions. The broader impact of our work includes reducing accident-related fatalities by identifying and mitigating key factors influencing accident severity. Additionally, our model can guide better road design and signage, potentially lowering insurance costs and overall vehicle ownership expenses.

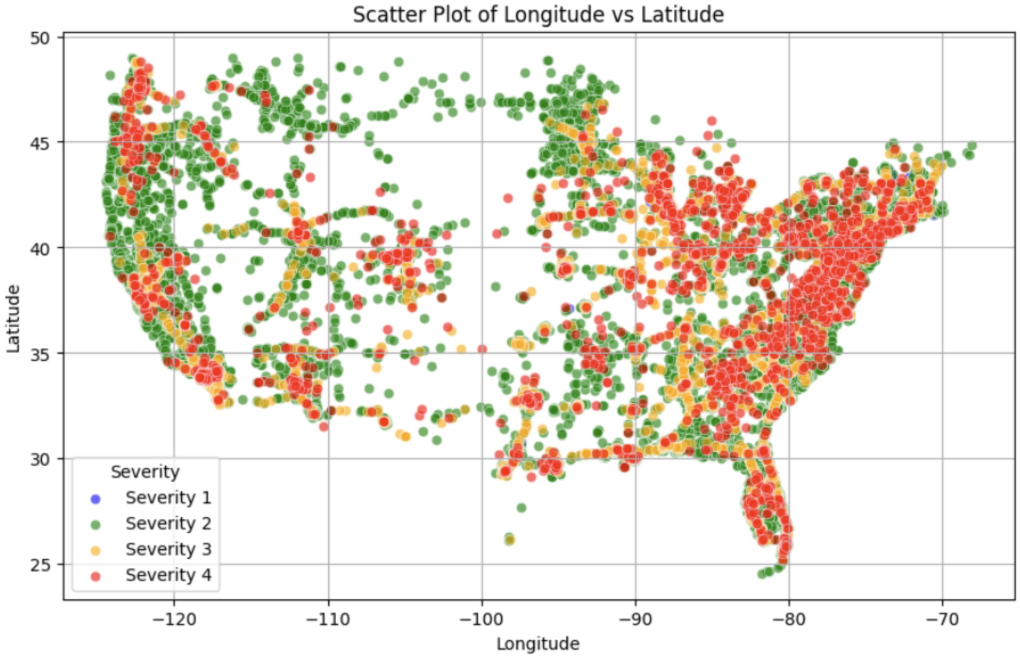


Figure 1: *Latitude vs. Longitude scatter plot, graphing the severity of reported accidents across the US*

1 Methods

1.1 Data Exploration

The dataset used in this study is the US Accidents dataset, covering the period from 2016 to 2023 and containing 7.7 million records across 49 states with 46 attributes [1][2]. The dataset was loaded using PySpark due to its large size. Initial exploration included examining the schema, checking for missing values, and generating basic statistics. A random sample of 1% was taken to create visualizations and perform further analysis using Pandas, Seaborn, and Matplotlib.

1.2 Pre-processing

Pre-processing steps involved removing redundant or irrelevant columns, extracting useful features from the timestamp, handling missing values, and encoding categorical variables.

1. The following columns were dropped: 'ID', 'Source', 'End_Lat', 'End_Lng', 'Timezone', 'Airport_Code', 'Weather_Timestamp', 'Street', 'City', 'County', 'State', 'Zipcode', 'Country', 'Turning_Loop', 'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight', 'Description', and 'Wind_Chill(F)'.
2. Numerical features were scaled using the Z-score method, and extreme values were removed based on a threshold.
3. Time-related features were extracted from the Start_Time column.

```
df['Start_Time'] = pd.to_datetime(df['Start_Time'])
df['Year'] = df['Start_Time'].dt.year
df['Month'] = df['Start_Time'].dt.month
df['Day'] = df['Start_Time'].dt.day
df['Hour'] = df['Start_Time'].dt.hour
df['Minute'] = df['Start_Time'].dt.minute
df['DayOfWeek'] = df['Start_Time'].dt.dayofweek
df = df.drop(['Start_Time', 'End_Time'], axis=1)
```

Listing 1: Python code to extract time-related features

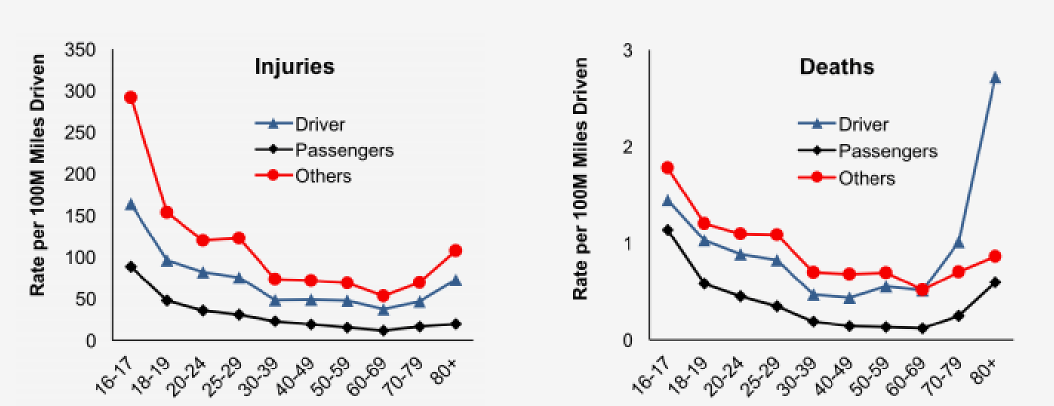


Figure 2. Injuries (left) and deaths (right) in crashes involving a driver of age shown per 100 million miles driven by drivers of that age, by role of person injured or killed, United States, 2014-2015.

Figure 2: *Injuries and deaths related to accidents in the US in the year 2014-15 [3]*

4. Categorical variables were encoded using one-hot encoding, and missing values were handled using median imputation for numerical columns and mode imputation for boolean columns.

```python
df['Sunrise_Sunset'] = df['Sunrise_Sunset'].map({'Night': 0, 'Day': 1})

weather_conditions = {
    'Clear': 'Clear',
    'Partly Cloudy': 'Cloudy', 'Mostly Cloudy': 'Cloudy', 'Overcast': 'Cloudy', 'Scattered Clouds': 'Cloudy', 'Cloudy': 'Cloudy',
    'Rain': 'Rainy', 'Light Rain': 'Rainy', 'Heavy Rain': 'Rainy', 'Showers': 'Rainy', 'Thunderstorm': 'Rainy', 'Drizzle': 'Rainy',
    'Snow': 'Snowy', 'Light Snow': 'Snowy', 'Heavy Snow': 'Snowy', 'Sleet': 'Snowy', 'Hail': 'Snowy',
    'Fog': 'Foggy', 'Mist': 'Foggy', 'Haze': 'Foggy',
    'Other': 'Other'
}
df['Weather_Condition'] = df['Weather_Condition'].map(weather_conditions).fillna('Other')

wind_directions = {
    'N': 'North', 'NNE': 'North', 'NE': 'North', 'ENE': 'North',
    'E': 'East', 'ESE': 'East', 'SE': 'East', 'SSE': 'East',
    'S': 'South', 'SSW': 'South', 'SW': 'South', 'WSW': 'South',
    'W': 'West', 'WNW': 'West', 'NW': 'West', 'NNW': 'West',
    'Calm': 'Calm', 'Variable': 'Variable'
}
df['Wind_Direction'] = df['Wind_Direction'].map(wind_directions).fillna('Variable')

df = pd.get_dummies(df, columns=['Weather_Condition', 'Wind_Direction'], drop_first=True)
```

Listing 2: Python code for one-hot encoding categorical variables

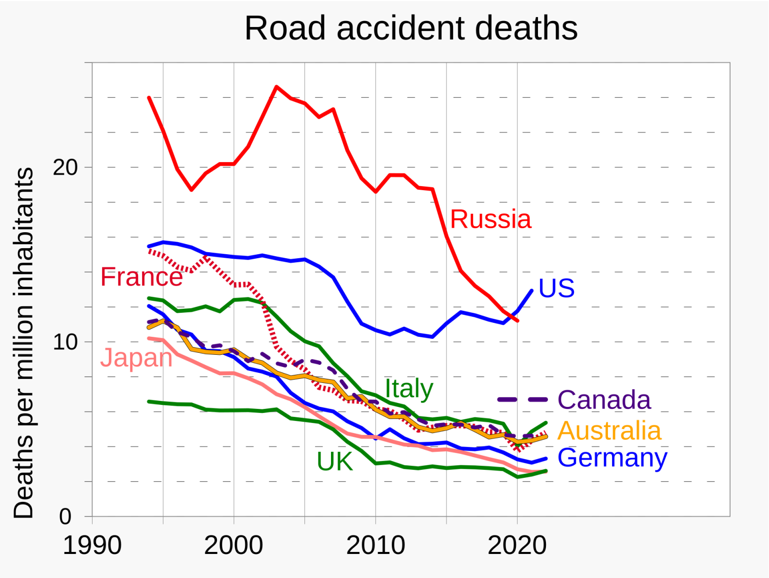


Figure 3: *Global accident statistics from 1998 - 2020 [4]*

1.3 Parameters

The final dataset included the following columns as features for model training:

- 'Start_Lat'
- 'Start_Lng'
- 'Distance(mi)'
- 'Temperature(F)'
- 'Humidity(%)'
- 'Pressure(in)'
- 'Visibility(mi)'
- 'Wind_Speed(mph)'
- 'Precipitation(in)'
- 'Amenity'
- 'Bump'
- 'Crossing'
- 'Give_Way'
- 'Junction'
- 'No_Exit'
- 'Railway'
- 'Roundabout'
- 'Station'
- 'Stop'
- 'Sunrise_Sunset'
- 'Year'
- 'Month'
- 'Day'
- 'Hour'
- 'Minute'
- 'DayOfWeek'
- 'Weather_Condition_Cloudy'
- 'Weather_Condition_Foggy'
- 'Weather_Condition_Other'
- 'Weather_Condition_Rainy'
- 'Weather_Condition_Snowy'
- 'Wind_Direction_East'
- 'Wind_Direction_North'
- 'Wind_Direction_South'
- 'Wind_Direction_Variable'
- 'Wind_Direction_West'

1.4 Model 1: Logistic Regression

The first model implemented was a logistic regression model. The model training and evaluation pipeline was created using the following libraries: `sklearn.model_selection` for splitting the dataset and performing grid search with cross-validation, `sklearn.linear_model` for logistic regression, `sklearn.preprocessing` for standard scaling, and `sklearn.metrics` for evaluating the model performance.

A pipeline was constructed to standardize the data and apply logistic regression. Grid search with cross-validation was used to find the best hyperparameters, optimizing for the macro F1 score.

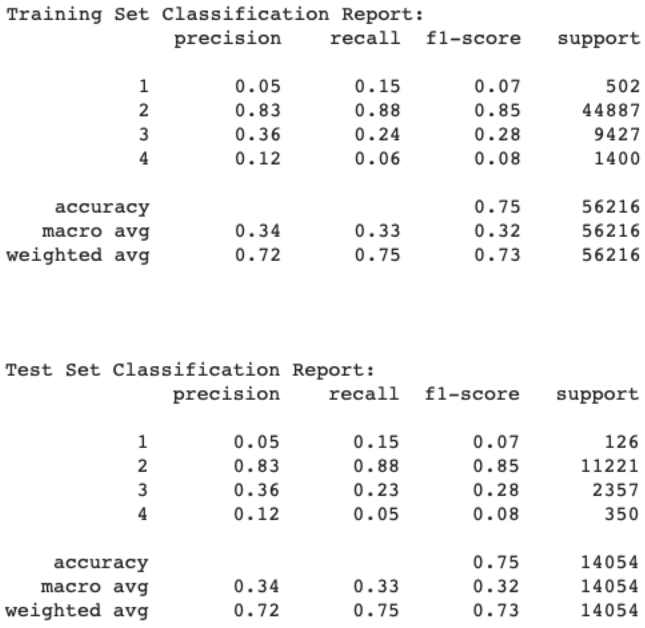
1.5 Model 2: Random Forest Classifier

The model training and evaluation pipeline was created using the following libraries: `collections` for handling class distribution, `imblearn.over_sampling` for SMOTE (Synthetic Minority Over-sampling Technique), `numpy` and `pandas` for data manipulation, `matplotlib.pyplot` and `seaborn` for visualization, `sklearn.model_selection` for splitting the dataset and performing grid search with cross-validation, `sklearn.ensemble` for the random forest classifier, `sklearn.metrics` for evaluating the model performance, and `sklearn.pipeline` for constructing the pipeline. To address class imbalance, SMOTE was applied to the training data. The dataset was then split into training and test sets, ensuring stratified sampling. A pipeline was constructed with the random forest classifier, and a grid search with cross-validation was used to find the best hyperparameters, optimizing for the macro F1 score. Model performance was evaluated on both the training and test sets, with classification reports, confusion matrices, accuracy scores, and F1 scores calculated.

2 Results

2.1 Model 1: Logistic Regression

The model achieved 74% accuracy on both the training and test sets, with F1 scores of 0.36, indicating no overfitting. The classification report for the model is as follows:-

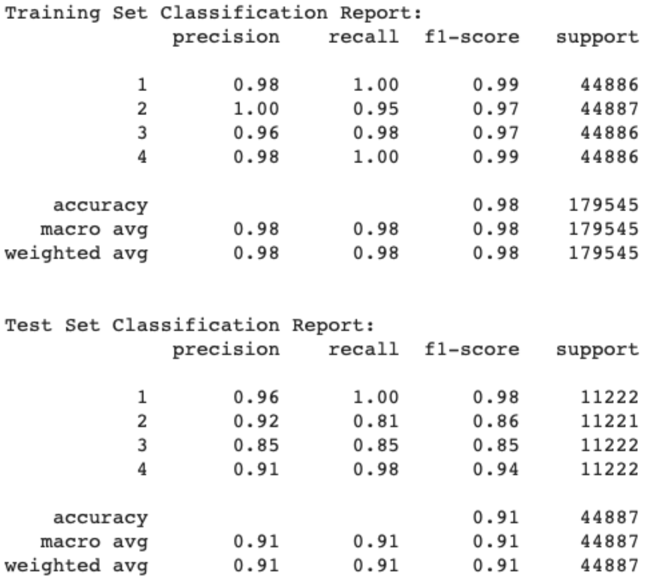


```
Training Set Classification Report:
              precision    recall  f1-score   support

           1       0.05      0.15      0.07       502
           2       0.83      0.88      0.85     44887
           3       0.36      0.24      0.28      9427
           4       0.12      0.06      0.08      1400

    accuracy                           0.75     56216
   macro avg       0.34      0.33      0.32     56216
weighted avg       0.72      0.75      0.73     56216



Test Set Classification Report:
              precision    recall  f1-score   support

           1       0.05      0.15      0.07       126
           2       0.83      0.88      0.85     11221
           3       0.36      0.23      0.28      2357
           4       0.12      0.05      0.08       350

    accuracy                           0.75     14054
   macro avg       0.34      0.33      0.32     14054
weighted avg       0.72      0.75      0.73     14054
```

Figure 4: *Classification reports of the logistic regression model on training and test sets*

2.2 Model 2: Random Forest Classifier

The model achieved 98% accuracy on the training set an accuracy of 91% on the test set. The classification report for the model is as follows:-



```
Training Set Classification Report:
              precision    recall  f1-score   support

           1       0.98      1.00      0.99     44886
           2       1.00      0.95      0.97     44887
           3       0.96      0.98      0.97     44886
           4       0.98      1.00      0.99     44886

    accuracy                           0.98    179545
   macro avg       0.98      0.98      0.98    179545
weighted avg       0.98      0.98      0.98    179545


Test Set Classification Report:
              precision    recall  f1-score   support

           1       0.96      1.00      0.98     11222
           2       0.92      0.81      0.86     11221
           3       0.85      0.85      0.85     11222
           4       0.91      0.98      0.94     11222

    accuracy                           0.91     44887
   macro avg       0.91      0.91      0.91     44887
weighted avg       0.91      0.91      0.91     44887
```

Figure 5: *Classification reports of the random forest classifier model on training and test sets*

3 Discussion

3.1 Data Exploration

In the data exploration phase, we utilized various techniques to gain insights into the structure and relationships within the dataset. A key tool used was the `heatmap`, which revealed significant correlations between variables. For instance, `Temperature` and `Wind_Chill` exhibited a very high correlation, indicating redundancy. Therefore, in the preprocessing stage, we decided to retain only one of these variables to avoid multicollinearity issues. Similarly, `Start_Lat` and `End_Lat`, as well as `Start_Lng` and `End_Lng`, were highly correlated, and since `End_Lat` and `End_Lng` had many missing values, we chose to keep only `Start_Lat` and `Start_Lng`. We also observed associations such as `Traffic_Calming` with `Bump` and `Traffic_Signal` with `Crossing`, suggesting that some `POI` (Point of Interest) annotations could be redundant or less relevant for predicting accident severity compared to weather conditions.

The scatterplot of accidents based on `latitude` and `longitude` provided a strong visual representation of accident distribution across the US, highlighting more populated areas with higher accident frequencies. This visualization underscored the importance of location-related features in modeling accident severity. To address the missing data issue, we used Spark functions to calculate the count of missing values across all columns, enabling us to identify which variables required imputation or could be excluded from further analysis. This comprehensive exploration phase laid a solid foundation for effective preprocessing and model building.

3.2 Pre-processing

The preprocessing phase involved several critical steps to clean and transform the data for better model performance. First, we focused on feature selection, removing columns that were redundant, not relevant for predicting severity, or highly correlated with other selected features. This step reduced dimensionality and potential noise in the dataset. Next, we engaged in feature engineering by extracting temporal features from the Start_Time column, such as Year, Month, Day, Hour, Minute, and Day_of_Week. These features helped capture temporal patterns that might influence accident severity, like rush hour traffic or seasonal variations.

We then simplified and encoded categorical variables. For example, Wind_Direction was grouped into broader categories like N, S, E, W, CALM, and VAR, and subsequently one-hot encoded. Similarly, Weather_Condition was converted into binary features for common weather conditions, and Sunrise_Sunset was encoded as a binary variable (0 for Day, 1 for Night). This approach ensured that categorical variables were represented in a format suitable for machine learning algorithms.

Handling missing values was another crucial step. For numerical columns, we used median imputation to fill missing values, as it is less affected by outliers compared to mean imputation. For categorical columns, we used mode imputation. This strategy preserved the distribution of the data while ensuring that no valuable data points were lost. Finally, feature scaling was applied using the StandardScaler to normalize numerical features (excluding Year), putting all features on a similar scale. This step is essential for algorithms sensitive to the scale of input data, improving their performance and convergence.

By meticulously addressing these preprocessing tasks, we ensured that the dataset was well-prepared for modeling, with reduced noise and enhanced feature representation, setting the stage for effective and reliable predictive modeling.

3.3 Logistic Regression

For the logistic regression model, we chose it due to its simplicity and interpretability, which makes it a good baseline for classification tasks. We utilized the LogisticRegression model from scikit-learn with a random state of 42 to ensure reproducibility and a maximum of 100 iterations to allow for convergence. The hyperparameter tuning grid included variations of the regularization parameter (C) to control overfitting, different penalties (l1 and l2) to manage regularization, solvers (liblinear and saga) suitable for small datasets and sparsity, and class weights to handle class imbalances.

The results showed that the model performed reasonably well on the majority class (severity 2) with high precision and recall. However, the performance on the minority classes (severities 1, 3, and 4) was poor, as indicated by lower precision, recall, and F1 scores. The confusion matrices revealed that the model struggled to correctly classify minority class instances, often misclassifying them into the majority class. This indicates that while logistic regression is straightforward and interpretable, its linear nature and sensitivity to class imbalance significantly limit its effectiveness for this dataset.

3.4 Random Forest Classifier

To overcome the limitations observed with logistic regression, we used a RandomForestClassifier. This ensemble method combines multiple decision trees to improve accuracy and robustness. We also used SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance by generating synthetic samples for the minority classes, resulting in a balanced dataset. The resampling led to an equal distribution of all classes, significantly improving the model's ability to learn from all categories.

For the model, we set a random state of 42 for reproducibility and a maximum depth of 20 for each tree to prevent overfitting. The hyperparameter grid included the number of estimators (n_estimators) and the number of features to consider (max_features) at each split. The Random Forest model demonstrated superior performance compared to logistic regression, achieving high precision, recall, and F1 scores across all classes. The confusion matrices showed a significant improvement in correctly classifying minority class instances, indicating the model's ability to capture complex patterns in the data.

The results from the Random Forest classifier were highly promising, with a substantial increase in accuracy and F1 scores for both the training and test sets. However, the model's complexity makes it less interpretable than logistic regression. Despite this trade-off, the Random Forest's robustness and ability to handle imbalanced datasets make it a preferable choice for this classification task.

4 Conclusion

Looking ahead, there are several avenues for potential improvement and further exploration. Implementing more sophisticated models such as Multi-Layer Perceptrons (MLP) or Support Vector Machines (SVM) could yield better performance by capturing non-linear relationships and intricate patterns within the data. MLPs, with their deep learning capabilities, can model complex interactions, while SVMs, known for their effectiveness in high-dimensional spaces, could provide a robust alternative.

Additionally, incorporating more advanced techniques for handling imbalanced datasets, such as ensemble methods combining oversampling and undersampling, or even using anomaly detection techniques to identify rare but significant accident cases, could enhance model performance further. Fine-tuning hyperparameters and exploring feature selection methods to reduce dimensionality without losing critical information would also be worthwhile.

In closing, this project not only highlighted the challenges and intricacies involved in predicting accident severity but also underscored the broader implications of such predictive models. Accurate prediction models can significantly contribute to road safety initiatives, helping to identify high-risk scenarios and informing preventive measures. By continuing to refine these models and integrating them with real-time data, we can move closer to reducing road accidents and injuries, ultimately saving lives and resources. The journey of this project reflects a small but meaningful step in the larger conversation on road safety and injury prevention, showcasing the power of data-driven insights in tackling real-world problems.

5 Statement of Collaboration

- **ISMAIL FAIZ**: Coder: Wrote all the code for the project. Reviewed the README's and the final report.
- **ARNAV TALREJA**: Writer: Responsible for all the README's as well as the final report. Provided feedback on the code during each milestone.

6 References

[1] Moosavi, S., Samavatian, M.H., Parthasarathy, S., & Ramnath, R. (2019). A Countrywide Traffic Accident Dataset.

[2] Moosavi, S., Samavatian, M.H., Parthasarathy, S., Teodorescu, R., & Ramnath, R. (2019). Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. Cambridge, MA: ACM.

[3] Tefft, Brian C. "Rates of Motor Vehicle Crashes, Injuries and Deaths in Relation to Driver Age, United States, 2014-2015 - AAA Foundation for Traffic Safety." AAA Foundation for Traffic Safety -, 14 June 2018, aaafoundation.org/rates-motor-vehicle-crashes-injuries-deaths-relation-driver-age-united-states-2014-2015/.

[4] "Motor Vehicle Fatality Rate in U.S. by Year." Wikipedia, Wikimedia Foundation, 19 July 2024, en.wikipedia.org/wiki/Motor_vehicle_fatality_rate_in_U.S._by_year.