Car Crash Prevention
Hotspot Identification and Prevention

Kevin Cha [1] **Jonathan Lam** [1] **Emily Zhou** [1]

Abstract

Our research question for this project is how to predict and prevent mobile vehicle crashes in Virginia, especially with crashes being a leading cause of death in the United States. By identifying hotspots for car crashes and the common causes for car crashes, we can work with state officials to better identify how to promote safer driving and reduce fatalities from car crashes.

1. Data

To answer our research question, we have used the VDOT's datatset on crashes within Virginia. https://dashboard.virginiadot.org/pages/safety/crashes.aspx

1.1. Dataset

This dataset consists of motor vehicle crash records in the state of Virginia since 2016. Each record corresponds to a different crash and provides detailed information on roadway conditions, driver factors, environmental conditions, location details, and affected individuals, along with the outcome of the incident. The raw dataset contains 69 variables and over 1 million observations. This dataset was chosen because it provided a comprehensive and reliable overview of all the different competing factors involved in a car crash. As such, we can better analyze how they influence the frequency and severity of collisions to build machine learning models that can assist government officials with public safety and maintenance decisions.

[*]Equal contribution [1]Department of Computer Science, University of Virginia, Charlottesville, VA, USA. Correspondence to: Kevin Cha <hpb2gv@virginia.edu>, Jonathan Lam <rgm6yp@virginia.edu>, Emily Zhou <csz6wd@virginia.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1.2. Key Variables

In order to build an effective model, we need to identify key variables to serve as features. Because we wish to discern the patterns behind car crashes, variables describing the severity and outcome of the accident, vehicles involved, environmental and weather conditions, road conditions, crash dynamics, traffic control, driver conditions, special populations, and whether it was work-zone-related are all included. Variable groups are listed instead of each individual variable due to the vast number of variables. Although driver conditions may not classify as an external contributing factor to a collision, context on whether a given driver was under the influence, distracted, drowsy, unbuckled, or speeding provides a more holistic understanding behind an accident's severity. Remaining variables containing metadata or identifiers are also dropped in the data cleaning pipeline.

These variables groups are key because they represent the broad domain of contributing factors in a collision. Variables pertaining to the crash severity and outcome serve as a baseline for predicting crashes, while variables about the vehicles involved (trucks, motorcycles, etc.) may also influence a crash's severity. Environmental and weather conditions have a direct effect on driving performance, and, consequently, the likelihood of an accident; heavy rain or snow limit visibility and influence reaction time. Road conditions play an equally impactful role, where potholes or icy roads could result in serious injuries. Crash dynamics and traffic control provide additional knowledge on an accident's severity. As aforementioned, driver conditions directly relate to incident severity, and special populations and work zone accidents highlight special at-risk groups and conditions.

1.3. Data Wrangling Challenges

The first challenge to handle with this dataset was the sheer size of the data from VDOT, as it had data from 2016 and had around 1 million entries. We wanted relevant data, considering our research question of what can be done to prevent crashes in Virginia. Take for example the case of roadway defects such as potholes, since they could have been fixed or new ones have emerged over time and use of the roads. The next step for data cleaning was to determine

which variables were still useful to our model, since while there were key variables, there were other variables such as OBJECTID which would not support our model so we dropped those columns completely.

The next big steps for cleaning the data were handling both categorical and numerical data, where we had to decide how to handle null values and also turn our categorical data into meaningful numerical data for our model to train on. For null values, it was determined that there were a few variables that only had a few null values so those rows were just dropped but for the following variables: Max Speed Diff, Functional Class, and Facility Type, had more than a few thousand null values and required further analysis. Max Speed Diff was only showing data where the difference between the speed of the car at the incident compared to the speed limit where the crash happened, so it was fair to fill out those null values with 0's. Then for Functional Class, it was a variable that determined where the crash occurred and since 10% of the values were NaN's, imputing them as "Unknown" sufficed. With the last variable with a significant number of null values being Facility Type, it also explained the format of the road where the crash occurred, so the remaining null values were imputed as "Unknown" as well.

With our categorical data, we had some issues discerning what the variables mean since there was no specific data dictionary for this dataset but after some thorough search the most confusing variables, K_people, A_people, B_people, C_people were determined to be on a scale referred to as KABCO. This scale indicated the damage to human lives, where K meant fatality, A meant serious injury, B meant minor injury, C meant possible injury, and O meant property damage only. The best way to handle this in a meaningful manner was to assign weight mappings to the variables, since K does have a higher severity due to lives being lost.

2. Methods

2.1. Method Overview

The task at hand is a classification task to predict the crash severity of an incident given various factors. The important features discussed previously are used with an 80:20 train/test split. The models used will be Random Forest, XGBoost, and Neural Networks. 5-fold cross validation will also be used to ensure that the model is not overfitting, and to optimally tune hyperparameters. The data will then be transformed by one-hot encoding the categorical features, and scaling the numerical features for stronger Neural Network performance.

2.2. Feature Filtering, Engineering, and Manipulation

After data wrangling, a few features were filtered, engineered, and manipulated to improve the model performance and avoid feeding junk features. Starting with the filtered features - any features with extremely low variance were dropped, since they are essentially constant and thus would not contribute meaningfully to the model's prediction aside from adding noise.

Existing features were also manipulated and engineered to produce features with less bins to reduce sparsity in certain categoricals. The following features were manipulated: coordinates, speed difference from limit, weather conditions, and light conditions/nighttime. Coordinates were modified by creating clusters using a KMeans cluster approach to create 100 area patches. The remaining features had some of their values binned and condensed - for example, for weather features, "rain", "drizzle", and "shower" were condensed to just "rain". This was done in cases where a feature may have had too many highly similar categories. A new feature was created by identifying cases where it was rainy and night, an expected combination of high crash risk.

These features were fed into a pipeline. This pipeline imputed any missing values. It also scaled the numericals with scikit-learn's StandardScaler, and one hot encoded the categoricals.

After these transformations were made to the data, further feature refinement was done by selecting the K best features with a mutual info classifier from scikit-learn.

Finally, the data was stratified, shuffled, and split via scikit-learn's StratifiedShuffleSplit. This was done to preserve sample ratios for the different classes when creating the train and test set.

2.3. Model Selection & Justification

Random Forest was selected as the first model choice for its robustness stemming from being an ensemble of Decision Trees. It typically performs well in classification tasks due to its capability to capture more complex relationships, which is important due to our large amount of features. Another strong suit of Random Forest is their interpretability, which allows for further analysis and understanding of what factors may influence crash severity.

XGBoost was selected as the second model choice due to its historically strong performance on classification tasks. Boosting as an ensemble learning method aims to optimize both bias and variance, so it may outperform Random Forest (a bagging ensemble learning method). Similar to Random Forest, there is strong interpretability from results as feature importance can be extracted.

A Neural Network is selected as the third model choice due

to its capability to handle very complex relationships thanks to the many nodes that can be added in hidden layers. Due to how many different features are present in the cleaned dataset, Neural Networks may outperform Random Forest or XGBoost given a more complex architecture capable of handling the features.

2.4. Model Training/Hyperparameters

Hyperparameter tuning was done for the Random Forest and XGBoost models via 5-fold cross validation.

For the Random Forest model, the following hyperparameters were considered and tuned: n_estimators, max_depth, and min_samples_split. n_estimators is an important hyperparameter because it tunes the number of trees to fit, which would lead to a more robust model for voting. max_depth is important to limit the depth of the tree and minimize overfitting. min_samples_split requires more samples for a category to justify a split, which is important to prevent overfitting on smaller less important categories in features. The result of the cross-validation found that the optimal hyper-parameters were the following: n_estimators = 100, max_depth = 75, and min_samples_split = 8.

For the XGBoost model, the following hyperparameters were considered and tuned: max_depth, subsample, γ, and min_child_weight. The max_depth parameter is tuned for similar reasons to the RandomForest model given that XGBoost is also a tree-based ensemble learning method and is equally important. The subsample hyperparameter forces subsampling of the training data for the trees, allowing for less overfitting and more robustness overall. The parameters γ and min_child_weight affect splitting behavior to further tune the tree optimally. The result of the cross-validation found that the optimal hyper-parameters were the following: γ = 0.5, max_depth = 5, min_child_weight = 1, and subsample = 0.6.

For the Neural Network, the final architecture contained two hidden layers, each with a dropout of 0.2 to prevent overfitting. It was optimized with the Adam optimizer. Training was completed with 60 epochs and a batch size of 256 at a time - early stop was also implemented with a patience of 10 epochs. The class weights were calculated to handle class imbalance.

2.5. Performance Metrics and Loss Function

The performance metric used for training and testing was F1 score. This is because in calculating crash severity the model should optimally maximize both precision and recall. Precision would measure how many crashes predicted as a severity rating were correct, while recall would measure how many crashes of a particular severity rating were correct. The model should perform well in both aspects. The loss function used is cross-entropy loss, as it is standard for classification tasks.

3. Results

3.1. Model Implementation and Training

Our initial model was a Neural Network implementation was a feedforward neural network with dense layers, ReLU, L2 regularization ($1e^{-4}$), and dropout (0.2). We had an Adam optimizer and utilized sparse categorical crossentropy loss, applied with 60 epochs with EarlyStopping. Additionally, we had class weighting and later bias corrections were attempted to address the severe class imbalance in the data itself yet this model still underperformed.

We then turned to the Random Forest and XGBoost models, which both had benefits for classification tasks as well as built in guardrails against overfitting. Our Random Forest model was tuned via grid search, we used balanced class weights, and extracted feature importances for interpretability. For our XGBoost model, it was tuned with around 1,000 estimators, monitoring on log-loss and classificatin error on validation set, and showed consistent improvement in $mlogloss$ and $merror$ over epochs.

3.2. Evaluation Benchmark

3.2.1. Dataset Benchmarking

The evaluation was performed on the 107,000 rows in our test set with all 5 severity classes represented. The issue was the the dataset was highly imbalanced, with the dominant class being the "Property Damage Only" class, being around ($\sim 68\%$) of the dataset. This resulted in recall on minority severe injuries particularly difficult, as seen in the varied attemps with various models.

3.2.2. Performance Metrics Used

Metrics used to evaluate these models are as follows:

- Accuracy
- Macro and weighted F1-score (crucial due to imbalance)
- Precision/recall per class
- Confusion matrices to examine misclassification patterns
- Our secondary analysis:
  - Geospatial heatmaps of predicted severity
  - Feature importance plots for Random Forest and XGBoost models

3.3. Initial Results and Comparisons

3.3.1. Neural Network Results

The raw test accuracy was 0.47 and even after tuning, the test accuracy was 0.53 with the tuned weighted F1 being 0.556. The macro F1 remained low due to poor performance on minority classes, despite attempts to address the class imbalance. The confusion matrix showed that there was severe overprediction of the majority class (Property Damage Only) and improved recall for Fatal situations due to bias adjustments but just extremely low precision (0.10), leading us to look at other models.

3.3.2. Random Forest Results

Our Random Forest model saw better successes, with the test accuracy being 0.705 and our weighted F1 was 0.631, demonstrating stronger overall performance on minority classes. As shown in Figure 1, feature importances shifted significantly after tuning. Certain road characteristics, weather codes, and location-based variables increased in importance. Unfortunately, it still struggled on minority classes, yet it still achieved notably better precision and F1 than the Neural Network model.

3.3.3. XGBoost Results

XGBoost saw the validation error decrease from 0.317 to 0.290 over the course of $1{,}000$ boosting rounds. As a result, the test weighted F1 was 0.6475, which was the best among all of the models. Additionally, the test accuracy was ~ 0.71. This particular model achieved exceptionally strong recall for the majority class (Property Damage Only), but recall remained very low for injury-related categories. As seen in the feature importance visualizations in Figure 2, XGBoost relied much more heavily on a subset of categorical roadway/environment features. The figures further show that while numeric geospatial features such as (x, y) were still meaningful, they were not dominant compared to the categorical roadway and environment features.

In Figure 3, we show clusters of high-severity predictions concentrated in specific geographic regions, suggesting spatial dependence.

3.4. Initial Result Analysis

3.4.1. Overall Trends

There was a ranking in models going from XGBoost, to Random Forest, to Neural Networks in both accuracy and F1. The Neural Network struggled substantially with not only high-dimensional ssparse categorical inputs from our VDOT dataset but also severe class imbalanced paired with struggles of capturing non-linear interactions without richer embedding-based preprocessing. Gradient boosted trees instead handled the structured tabular data much more effectively.

3.4.2. Class Imbalance Effects

All models performed poorly on Fatal, Severe Injury, and Visible Injury classes despite efforts with class weighting and bias correction. The minority classes still had low precision and recall across all models and most notably did the Neural Network model just dramatically overpredict the majority class but improved the Fatal recall at the cost of precision.

3.4.3. Interpretability Insights

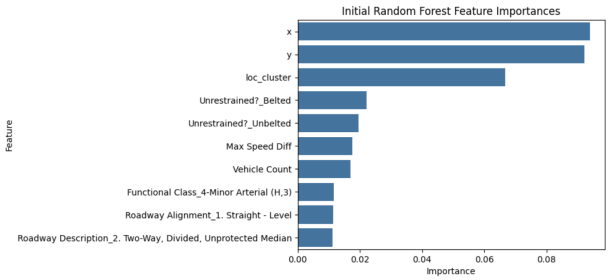
Our models for Random Forest and XGBoost brought light to similar key predictors such as Roadway type, Reporting source, Light/weather conditions, Intersection-related activities, and Geospatial clustering (via heatmaps). All of the following features do align with known crash severity determinants from transportation safety literature, indicating that the tuning was effective and the models were in fact using the most applicable features to calculate crash severity as effectively as possible.

4. Conclusion

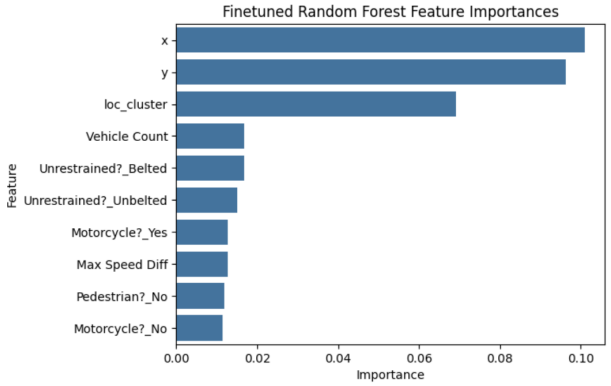
4.1. Study Summary

The goal of this research was to identify vehicle crash hotspots within Virginia by predicting their crash severity using the VDOT crash dataset, which detailed environmental, roadway, driver, and situational attributes for over one million crashes. Crash severity followed the KABCO scale, a nationally recognized model developed by the U.S. Department of Transportation Federal Highway Administration (FHWA) that categorizes crashes into individuals killed (K), serious injury (A), minor injury (B), possible injury (C), and property damage only (O). When developing our machine learning models, each of these classes were treated as a distinct category for a multi-class classification problem. To prepare our dataset for modeling, we identified key variables, cleaned missing values, transformed categorical variables for our models to interpret, and followed a KMeans cluster approach to condense our feature space for highly similar categories. The resulting data were fed into a pipeline that imputed values where necessary, scaled numerical values, one-hot encoded categorical variables, and stratified, shuffled, and split the data before being passed to our models.

Three models were evaluated: a Random Forest classifier, an XGBoost gradient boosting model, and a feedforward Neural Network. Model performance was evaluated with accuracy, precision/recall per class, and macro and weighted F1 score. The Neural Network struggled with sparse, high-

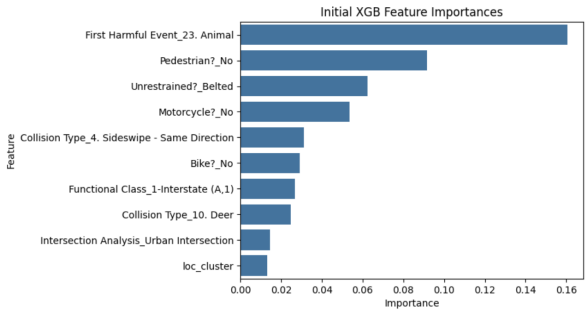


(a) Initial Random Forest Model's Feature Importances

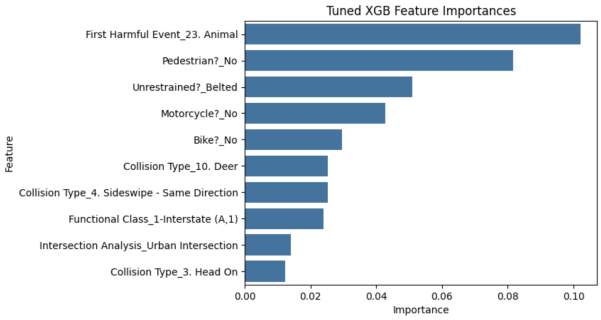


(b) Tuned Random Forest Model's Feature Importances

Figure 1. Random Forest feature importances before and after hyperparameter tuning. Tuning altered the model's reliance on key roadway, environmental, and location-based variables, with several categorical roadway attributes and lighting/weather conditions emerging as more influential predictors of severity. This shift highlights the sensitivity of tree-based models to hyperparameter optimization.



(a) Initial XGBoost Model's Feature Importances



(b) Tuned XGBoost Model's Feature Importances

Figure 2. Initial and tuned XGBoost feature importances. The tuned model places substantial weight on a subset of roadway characteristics, environmental factors, and high-cardinality categorical features. Compared to the Random Forest model, XGBoost demonstrates sharper prioritization, reflecting its ability to exploit complex nonlinear interactions within structured tabular crash data.

cardinality inputs and class imbalance, producing the lowest overall performance despite hyperparameter tuning and class-weight adjustments. On the other hand, the Random Forest model produced a stronger accuracy and weighted F1 score, for it handles class imbalance and nonlinear interactions more robustly. The XGBoost model performed best overall with an accuracy of 0.71 and the highest weighted F1 score of 0.6475. Feature importance rankings were also performed for each model, as well as a geospatial analysis of predictions to reveal the distribution of high-severity crash clusters across the state.

By analyzing our feature importance rankings and crash heatmap, we can identify geographic areas of high-risk crashes and their most consistent risk factors, providing comprehensive answers to the initial research question. Despite these successes, however, all models struggled with predicting the Fatal and Severe Injury crashes due to extreme class imbalance. Future improvements should focus on addressing this class imbalance through techniques like SMOTE or strategic under sampling, exploring richer geospatial modeling techniques such as graph-based road networks, embedding layers for categorical variables in our Neural Network model, and ensuring each models' confidence scores match real-world frequencies within the VDOT data. Moving forward with these improvements, we hope to benefit public safety officials by translating model predictions into actionable strategies for reducing crash severity across the state.

4.2. Defending from Criticism

While this study provides an analysis of both the importance of differing factors when it came to crash severity (location, driver details, weather conditions, etc.) and the capability to predict crash severity in Virginia, it's important to recognize that the cleaning decisions, data used, models utilized may have their own inherent shortcomings or biases. To respond to potential criticisms, it's important to re-emphasize and highlight decisions made during the methodology of the

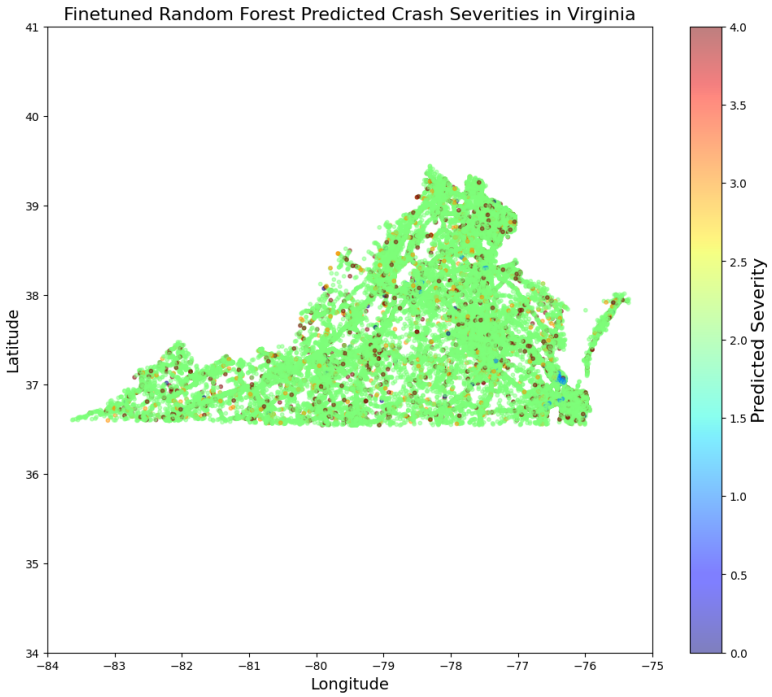


Figure 3. Spatial heatmap of predicted crash severity across the study area. Red regions correspond to higher predicted severity classes, while blue regions represent lower severity predictions. The geographic clustering of high-severity predictions suggests strong spatial dependence in crash outcomes, particularly along major road corridors and high-traffic intersections.

Table 1. Comparison of model performance on the VDOT crash severity prediction task.

Model	**Accuracy**	**Weighted F1**	**Macro F1**	**Notes**
Neural Network (Raw)	0.4703	0.520	0.31	Baseline, no bias correction
Neural Network (Tuned)	0.5326	0.5563	0.31	Class weights + bias adjustment
Random Forest (Tuned)	0.7051	0.6314	–	Best non-boosting baseline
XGBoost (Tuned)	0.7100	0.6475	0.36	Best overall model

experiment.

The selection of what factors to include was empirically decided based on relevant factors to crashes (and more generally, driving in general). Thus, administrative data and data that was duplicated such as VDOT district, presence of Virginia State Police, Document number, etc. was removed. Another decision made was with respect to manipulation of features to be binned and condensed - this is important as rare categories can cause issues with decision boundaries (especially for tree-based models). Lastly, feature selection with KBest was performed for dimensionality reduction and to prevent overfitting of the model and was not biased by human selection. Thus, decisions related to feature selection was unbiased and completed algorithmically.

It's also important to understand and analyze that there may be inherent issues with the data itself. With regards to the quality and completeness of crash-related information and features, only null information was dropped or imputed. Given that this is a state-level approved dataset provided by the Virginia Department of Transportation, there were no concerns with the given data. The analysis accounted for these issues. The data also does not encompass crashes that were unreported, this results in a potential gap (especially in lower crash severity incidents) but does not constitute an issue for the purpose of this study.

It has also been acknowledged that there was class imbalance which affects model performance and by association, feature importance, but the decision to use Random Forest and tuning appropriate hyper-parameters reflects attempts to mitigate this issue (which may be an explanation for Random Forest's stronger performance). Furthermore, regardless the imbalance observed across severity levels reflects reality as higher crash severities tend to be rarer.

4.3. Potential Improvements and Next Steps

Evidently, one of the biggest issues all three models ran into was the severe class imbalance that exists in the dataset. A common solution is applying SMOTE, which is an oversampling approach, or instead using undersampling to address the imbalance. If the next steps applies SMOTE, the iteration after that should implement hybrid approaches such as SMOTE-Tomek or SMOTEENN could reduce the noise introduced during oversampling. Another approach would be applying cost-sensitive learning, such as focal loss or class-specific penalty weights, would further force the Neural Network models to pay greater attention to Fatal and Severe Injury cases. With all of this in mind, if we chose to revisit and iterate upon on our Neural Network model a bit further, implementing these approaches would definitely pinpoint the differentiating factors for performance between our Neural Network model in comparison to our other two models.

Another approach for this problem as a whole would be to considering further exploring Geospatial modeling considering the nature of the dataset and the work at hand. Some geospatial modeling techniques that could have been applied to improve upon this project would be including geohash-based features, spatial smoothing, and graph-based road network features. If we were to implement geospatial modeling, we could take it one step further. Since crashes on adjacent road segments are not independent, incorporating spatial autocorrelation tests or graph-based models (for example, GNNs on road network graphs) could substantially improve and preserve the realism of severity predictions, which is key for research similar to this project. On the topic of realism, a huge part of this dataset beyond the geospatial aspects is the temporal one. For temporal data, it greatly dictates the driver activities on Virginian roads, so incorporating temporal structure could improve prediction of conditions that change over time, such as seasonal weather risks or rush-hour patterns thus preserving the realism of our predictions.

The final potential improvement would be considering the evaluation of the calibration of the severity probabilities to improve the usefulness of the decision, which would benefit all models. Well-calibrated outputs would ensure that predicted severity levels correspond to real-world crash frequencies, which allows transportation officials to use these models not only for classification but for informed decision making. Altogether, these options for future directions to take this project into illuminate the broader significance of our work. While our models, particularly XGBoost, demonstrated promising performance, their limitations point to clear pathways for refining accuracy and practical utility. By addressing imbalance, deepening spatial and temporal modeling, and improving calibration, this research can meaningfully assist public safety efforts and help guide targeted inventions to reduce crash severity across Virginia.

References

[1] Brownlee, J. (2021, March 17). SMOTE for Imbalanced Classification with Python. MachineLearningMastery.com. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[2] Hyperparameter tuning. GeeksforGeeks. (2023, December 7). https://www.run.ai/guides/hyperparameter-tuning

[3] Koehrsen, W. (2018, January 9). "Hyperparameter Tuning the Random Forest in Python Using Scikit-Learn." https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

[4] Lee, E. (2023, April 11). Hyperparameter optimization: Grid search vs. Random Search vs. Bayesian Optimization in Action. Medium. https://drlee.io/hyperparameter-optimization-grid-search-vs-random-search-vs-bayesian-optimization-in-action-106f99b94e32

[5] Run.ai. (n.d.). Hyperparameter tuning. Hyperparameter Tuning: Examples and Top 5 Techniques. https://www.run.ai/guides/hyperparameter-tuning

[6] Virginia Department of Transportation. (2024, December 8). VDOT Dashboard: Crashes. https://dashboard.virginiadot.org/pages/safety/crashes.aspx