# Traffic Crash Prediction Using Machine Learning

Group Members: Vanessa W., David S., and Amidou O.

# Introduction

This project focuses on using historical traffic and environmental data to predict the likelihood of road accidents at specific locations and times. By analyzing patterns related to factors like traffic volume, road conditions, weather, and time of day, the predictive model will identify “hotspots” and peak hours for crashes. Armed with these insights, city planners and transportation officials can take proactive steps—such as improving infrastructure, adjusting traffic signals, or issuing targeted warnings—to enhance road safety and reduce the number of accidents.

# Walk through your method

**Our Goal:** To build a machine learning model that predicts the severity of a traffic crash based on various features like location, time, and environmental factors.

**1. Data Exploration and Preprocessing:**

- We first imported necessary libraries like pandas, NumPy, and scikit-learn for data manipulation, analysis, and modeling.
- We then loaded the traffic crash dataset and examined its shape, columns, missing values, and data types using functions like df.shape, df.columns, df.isnull().sum(), and df.dtypes.
- We handled missing values by dropping columns with excessive missing data and imputing missing values in other columns using median (for numerical features) and mode (for categorical features).
- We extracted time-based features (year, month, day, hour) from the 'DateOfCrash' column to capture potential temporal patterns in crash severity.
- We encoded categorical features using Label Encoding to convert them into numerical representations suitable for our machine learning model.
- We also created a 'Crash Severity' category based on injury information, classifying crashes into different severity levels (fatal, severe, minor, no injury).

**2. Feature Engineering and Selection:**

- We explored correlations between features and the target variable ('Crash Severity') using df.corr(). This helped us identify potentially important predictors and understand feature relationships.
- We made a crucial decision to remove features directly related to injury outcomes (like 'killed', 'type_a_injury', etc.) to prevent data leakage and ensure our model predicts severity based on contributing factors rather than direct injury information.
- We identified and removed some features that were highly correlated or redundant to avoid multicollinearity issues and improve model generalization.

**3. Addressing Class Imbalance:**

- We recognized that the dataset likely had class imbalance, where certain crash severity levels were much more frequent than others.
- We used the SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples for the minority classes, balancing the dataset and improving our model's ability to learn from all severity levels effectively.

**4. Model Selection and Training:**

- We chose a Random Forest Classifier as our model due to its robustness in handling imbalanced data, its ability to capture complex relationships, and its interpretability.
- We trained the Random Forest model on the resampled training data using rf.fit(X_train_resampled, y_train_resampled).
- We tuned the model's hyperparameters (n_estimators, max_depth, min_samples_split) to find optimal settings for our dataset.

**5. Model Evaluation and Interpretation:**

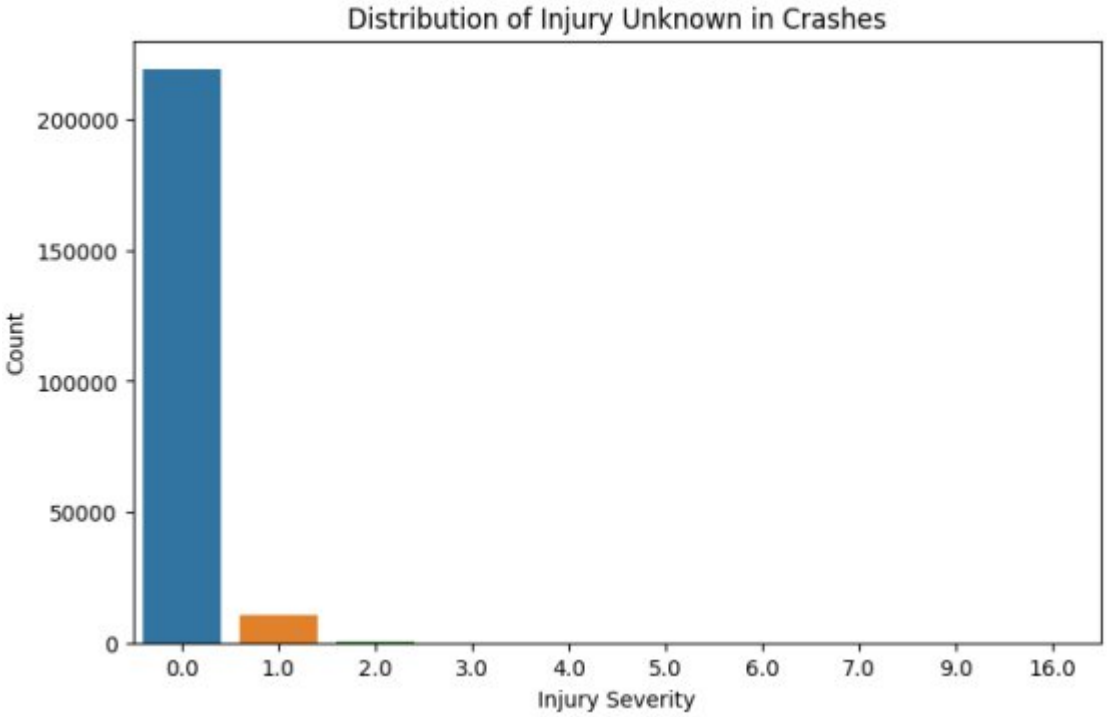
- We evaluated the model's performance on the test data using metrics like accuracy and classification report.
- We visualized the feature importances generated by the Random Forest model to understand which features were most influential in predicting crash severity.
- We iteratively refined our feature set and model parameters to improve its accuracy and generalization capabilities.

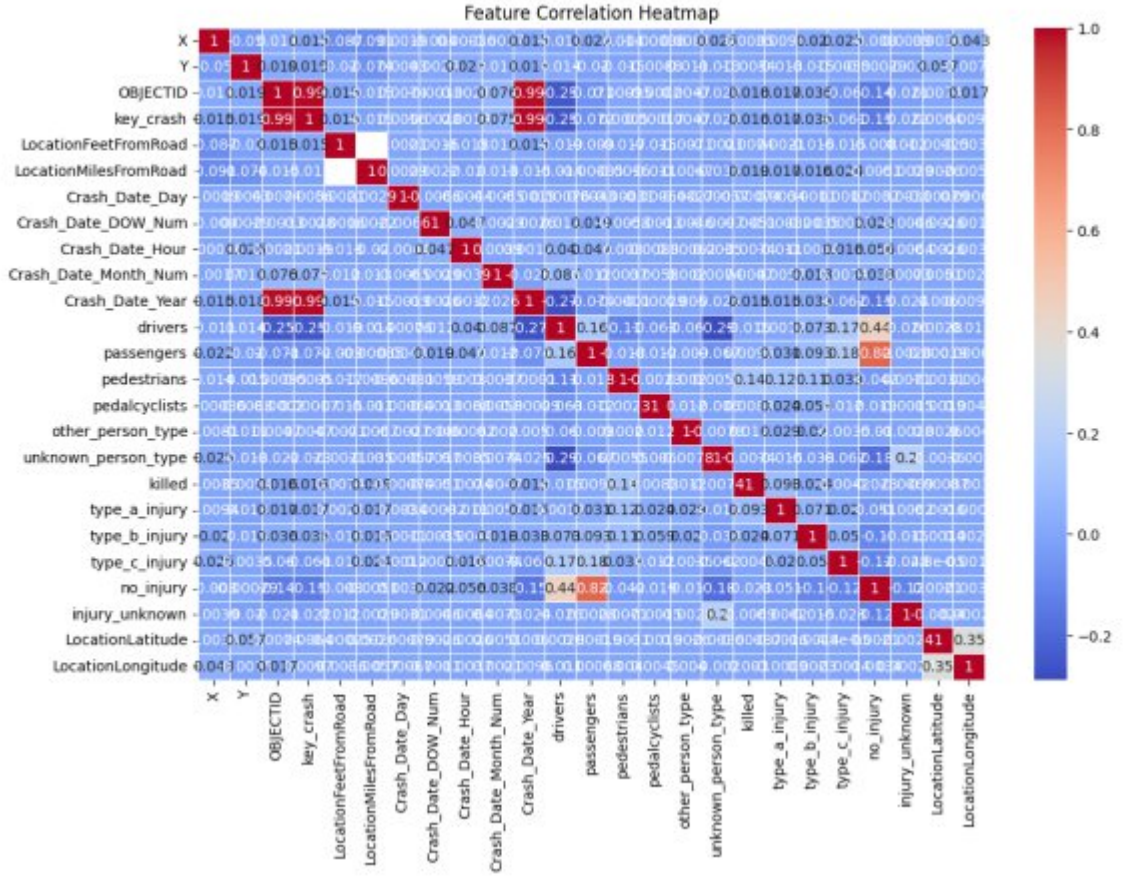
- We analyzed the classification report to assess the model's performance across different severity classes, focusing on precision, recall, and F1-score, particularly for minority classes.

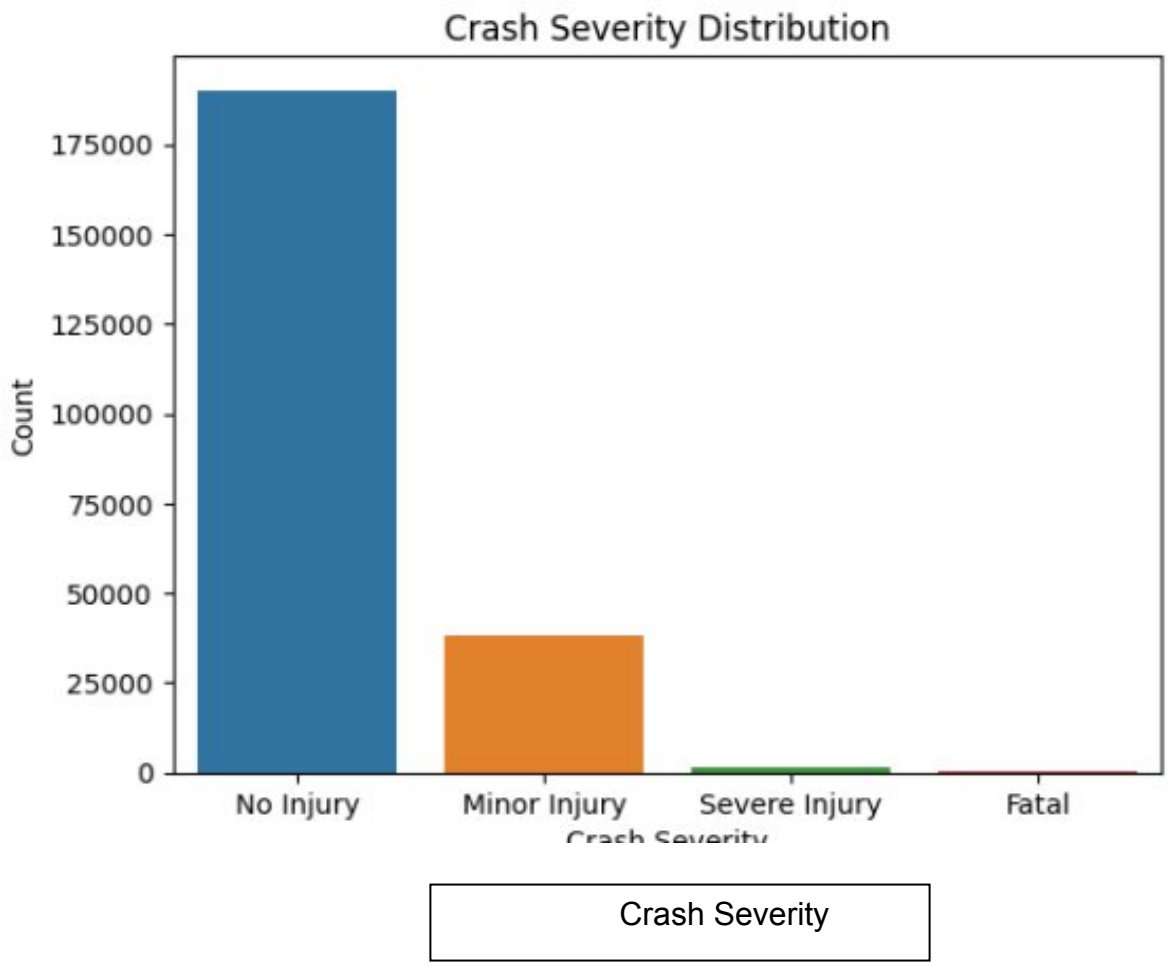
**In Summary:**

We followed a structured approach to this project, encompassing data preprocessing, feature engineering, handling class imbalance, model selection, training, and evaluation. This allowed us to develop a model that can potentially help predict traffic crash severity based on a variety of factors. This project was also visualized into a steamlit application with limited functionality and user interaction

## Appendix : Important fields like injuries and fatalities have some missing values but are usable


Distribution of Injury Unknown in Crashes
Count
200000
150000
100000
50000
0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
9.0
16.0
Injury Severity


Feature Correlation Heatmap
X
Y
OBJECTID
key_crash
LocationFeetFromRoad
LocationMilesFromRoad
Crash_Date_Day
Crash_Date_DOW_Num
Crash_Date_Hour
Crash_Date_Month_Num
Crash_Date_Year
drivers
passengers
pedestrians
pedalcyclists
other_person_type
unknown_person_type
killed
type_a_injury
type_b_injury
type_c_injury
no_injury
injury_unknown
LocationLatitude
LocationLongitude
1.0
0.8
0.6
0.4
0.2
0.0
−0.2




**What We've Learned and Further Questions:**

1. **Data Preprocessing:** We learned how to handle missing values and encode categorical features. However, we're still curious about other imputation techniques like KNN or iterative imputation. *Could exploring these alternatives improve model accuracy?*
2. **Feature Engineering:** Creating time-based features was insightful. We wonder if engineering features related to weather or road conditions could enhance the model's predictive power. *Where can we find reliable external datasets to incorporate such features?*
3. **Class Imbalance:** SMOTE effectively balanced the dataset, but we're interested in exploring other techniques like undersampling or different SMOTE

variations. *Could these alternatives lead to better model performance, particularly for minority classes?*

4. **Model Selection:** We used a Random Forest Classifier, which performed well. However, we're curious about comparing it to other models like XGBoost or Support Vector Machines. *Would these alternatives offer better performance or interpretability?*
5. **Hyperparameter Tuning:** While we tuned some hyperparameters, we're curious about using more advanced techniques like Bayesian Optimization or genetic algorithms. *Could these methods lead to finding even better hyperparameter settings?*
6. **Feature Importance:** Analyzing feature importance was helpful. We're interested in exploring techniques like SHAP values or permutation importance for a deeper understanding. *Could these methods reveal more nuanced insights into feature contributions?*
7. **Model Interpretability:** While Random Forests offer some interpretability, we're curious about exploring methods like LIME or decision tree visualization to explain individual predictions. *How can we make the model's predictions more transparent?*
8. **Deployment:** We're interested in learning how to deploy this model as a web application or API. *What tools and frameworks are suitable for deploying machine learning models in a production environment?*
9. **Data Drift:** We're curious about how to monitor for data drift and retrain the model to maintain its accuracy in real-world scenarios. *What strategies can be employed to handle data drift and ensure model reliability over time?*
10. **Ethical Considerations:** This model could be used to inform decisions. We're interested in exploring the ethical implications and how to mitigate potential biases. *How can we ensure the responsible and equitable use of this model?*