ST595 – Transportation Engineering with CAVs
Spring 2025
Project # 2: Compare Avs and HVs

Dezhong Xu

For this project you will create a pdf or HTML file with text, code, and results (perhaps using an R Markdown output, a Jupyter Notebook, or a `.qmd` file output). This output file (pdf or HTML) and the corresponding file used to create the output (`.Rmd`, `.ipynb`, or `.qmd`) should then be uploaded to wolfware in the assignment link!

Goal

The purpose of this project is to create a report where you read in data, fit models using concepts from the class, and choose between several different models.

Data (Must be Approved by 3/26!)

- Find a data set you can fit supervised learning models with
    - You cannot use the datasets we've been using in class
    - You can't use a super simple data set such as the iris data set nor data on wine quality.
    - There are many places with free data out there such as the UCI machine learning repository and kaggle
- You can focus on a classification task or a regression task, I'll leave that to you!
- Please read your data in via a URL or include the data as a file in your submission.
- Please send me an email with a link to the data ASAP and I will let you know if the data is OK to use for the project.

To Do:

Create a document that goes through your process of reading the data, any basic data cleaning/transformations, splitting the data, and fitting and choosing a final model. Details are given below.

Read in the Data

- Give a brief introduction to the data and the source of the data.
- State your goal for the project (modeling something). Be specific on why you want to model the variable.
- Read the data in via a URL (or locally but then you must submit the data with your submission)

Data Cleaning & Transformations

- Go through a brief process where you make sure that each variable is read in correctly (correct data type), check for missing values, check for valid data values (perhaps by summarizing each column and seeing if you have reasonable values), removing observations where needed (give an explanation as to why), and doing any data transformations you deem necessary.

Split the Data into a Train and Test Set

- Use whatever reasonable proportion you'd like.

Training Models

In the course so far, some of the models we've looked at are

- kNN
- (Regularized) Linear Regression & Logistic Regression Models
- GAMs (using piece-wise polynomial regression, local regression, or splines)
- Single tree models
- Ensemble tree models
- Support vector machines

You'll fit one model from each of these model types (I'll let you determine the details).

For **each model type**, do the following:

1. Describe the model. For instance,
    - is the model a parameteric model? non-parametric?
    - does it have tuning parameters? if so, what are they?
    - can we use the model for inference?
    - does the model perform variable selection?
    - do we need to standarize the predictors?
2. Fit the models on the training set, selecting tuning parameters where needed using a reasonable metric.
3. Select the 'best' model and refit it to the training data.

Testing Models

Test each of your models on the test set. Report appropriate metrics and discuss which model you prefer and why (perhaps simplicity, perhaps predictive ability, etc.)

Fit your overall best model on the entire data set.

Interpret this model as best you can. That is, if you can do inference with the model, do so. If you can simply report variable importance measures, do that. If you can produce plots that describe the relationships, do that!

That's it! Submit the appropriate files.

1. Introduction

1.1 Data Source

The dataset used in this project is an open-source traffic accident dataset available on Kaggle. It contains detailed records of traffic accidents across various regions and time periods, with a total of 209,307 rows and 24 columns. The dataset includes a wide range of quantitative and qualitative variables, such as accident date, weather conditions, lighting conditions, crash types, and vehicle involvement. These attributes provide valuable insights into traffic incidents and their potential causes.

The data could be accessed through the link below:
https://www.kaggle.com/datasets/oktayrdeki/traffic-accidents/data.

1.2 Scope of the Project

Building on the information provided by the dataset, the scope of this project is to develop a robust inference model that explores the relationship between environmental and roadway conditions and the severity of intersection traffic accidents.

The decision to focus on predicting accident severity—rather than the number of injuries—is intentional. Although injury count is available, it does not provide reliable insights on an incident-by-incident basis, as the number of passengers involved in each accident is neither standardized nor predictable, especially without information on vehicle types. In contrast, accident severity offers a more consistent and interpretable target variable for modeling.

Predicting injury counts would be more appropriate in a different context, such as when analyzing data from a single location over a longer time frame (e.g., annually). However, such an analysis lies outside the scope of this project.

Furthermore, because the dataset includes accident records across diverse regions and time periods, it is not practical or intuitive to build a model that attempts to explain the causes of accident severity across all transportation facility types simultaneously. Instead of constructing separate models for each of the 20 facility types identified under the 'trafficway type' variable, this project will focus solely on intersection-related traffic accidents. The goal is to identify how factors such as traffic control methods, lighting conditions, crash types, and weather influence accident severity specifically at intersections.

2. Data Preparation

2.1 Data Screening and Variable Pre-selection

Before developing the traffic accident severity prediction model, it is crucial to carefully curate both the dataset and the input variables. The objective is to retain only those features that are meaningful and relevant to accident severity, while excluding variables that are unrelated, redundant, or weakly correlated.

As outlined in the project scope, this study focuses specifically on intersection-related accidents. To align the data with this focus, a filtering process was applied to include only records corresponding to intersection-type trafficways. Specifically, entries labeled as 'Four-Way,' 'T-Intersection,' 'L-

Intersection,' 'Y-Intersection,' and 'Roundabout' under the trafficway type variable were retained. This filtering reduced the dataset to 59,524 records.

Several quantitative techniques can assist in the variable selection process. For example, statistical methods such as ANOVA can evaluate the significance of each variable through p-values, helping identify those that strongly influence the target variable. Additionally, regularization methods like LASSO regression provide built-in variable selection by penalizing less important features.

However, beyond quantitative techniques, an initial qualitative screening based on domain knowledge is equally important. This helps ensure that irrelevant or redundant variables—which may introduce noise or reduce model interpretability—are excluded early in the process.

Although the original dataset contains 24 columns, concerns about the curse of dimensionality and the need for model rationality led to the exclusion of repeated or irrelevant variables. The final set of selected features used for modeling is introduced below.

Time variables

Crash_hour: The hour the accident occurred.

Crash_day_of_week: The day of the week the accident occurred.

Crash_month: The month the accident occurred.

Condition variables:

traffic_control_device: The type of traffic control device involved (e.g., traffic light, sign).

weather_condition: The weather conditions at the time of the accident.

lightining_condition The lighting conditions at the time of the accident.

first_crash_type: The initial type of the crash (e.g., head-on, rear-end).

alignment: The alignment of the road where the accident occurred (e.g., straight, curved).

roadway_surface_condition: The condition of the roadway surface (e.g., dry, wet, icy).

road_defect: Any defects present on the road surface.

Response Variable:

most_severe_injury: The most severe injury sustained in the crash.

2.2 Data Cleaning and Transformations

Given that the dataset contains a substantial number of qualitative variables, it is essential to ensure that each level within these variables is sufficiently represented. Inadequate sample sizes for certain levels can lead to biased model estimates and, as a result, misinterpretation of model parameters.

For example, the variable traffic_control_device contains 18 distinct levels. However, the distribution of observations across these levels is highly imbalanced. Specifically, intersections controlled by a 'NO PASSING' sign are represented by only 3 records, while 'SCHOOL ZONE' and 'RAIL ROAD CROSSING SIGN' appear in just 6 and 9 records, respectively. Levels with such low sample sizes are unlikely to contribute meaningfully to model training and may introduce noise or instability in parameter estimation.

To enhance the reliability and interpretability of the model, these sparsely populated levels were excluded from the dataset. The threshold for exclusion was set objectively at 1% of the total sample size—equating to approximately 595 observations in this case. Only levels with a sample size exceeding this threshold were retained for modeling.

In addition, several qualitative variables included a level labeled ‘UNKNOWN’, which typically reflects incomplete or ambiguous accident records. These ‘UNKNOWN’ levels were also removed to ensure the clarity and integrity of the modeling process. The response variable ‘most_severe_injury’ also contains a fuzzy level ‘REPORTED, NOT EVIDENT’ which indicate no injury level reported. So, any data with that variable level will be excluded.

Following the filtering process, the cleaned dataset was reduced to 39726 observations. Upon further inspection, it was found that the ‘road_defect’ contained only a single level 'NO DEFECTS'. Since variables with no variability do not contribute to the predictive power of a model, both were removed from the dataset prior to modeling.

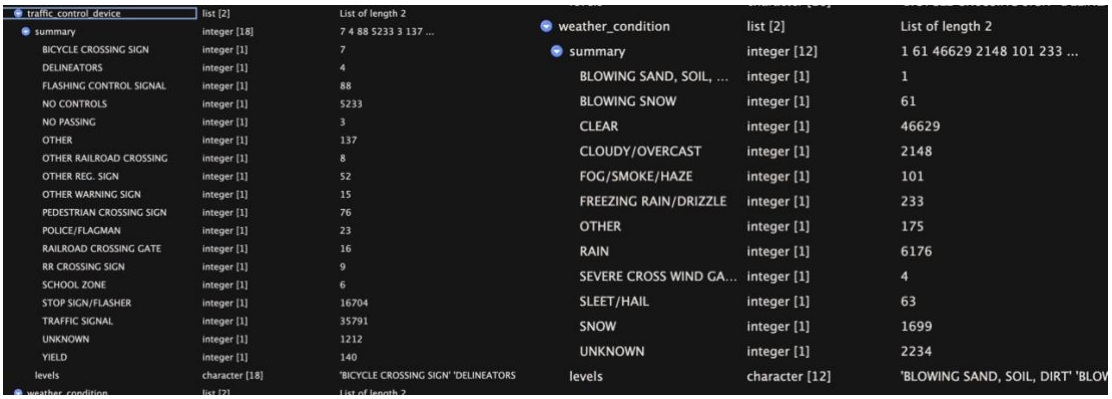

Figure 2-1- Traffic Control Device and Weather Condition Levels before Data Cleaning

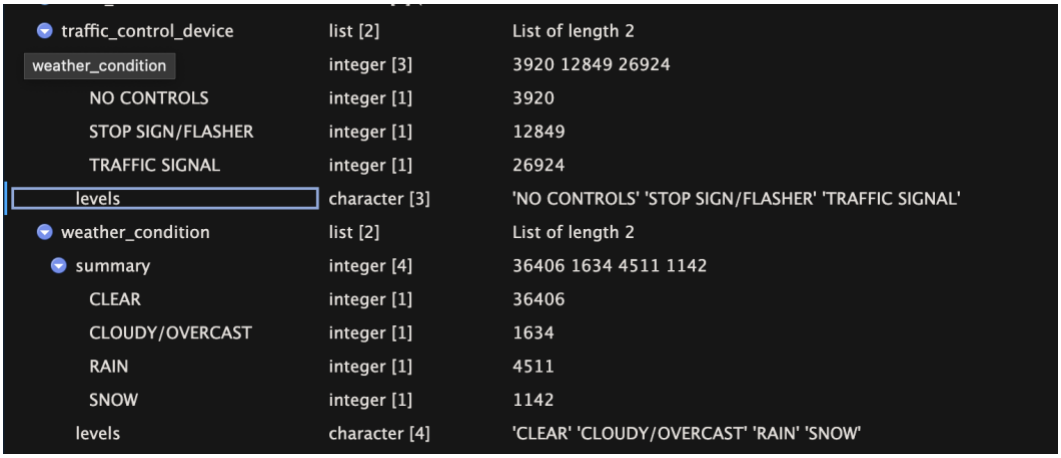

Figure 2-2 - Traffic Control Device and Weather Condition Levels after Data Cleaning

After data cleaning, a transformation was applied to the time-spatial variables to enhance interpretability. The original variables included crash hour, day of the week, and month. However, from a traffic engineering perspective, this level of detail can be overly granular for meaningful analysis. For instance, comparing traffic patterns on an hour-by-hour basis may not reveal significant differences

between adjacent time windows. Therefore, crash hours were reclassified into broader categories such as AM peak, PM peak, off-peak, and nighttime hours, which better reflect typical traffic flow patterns.

Similarly, the days of the week were grouped into two categories: weekdays and weekends, to capture broader behavioral and traffic trends. Month information was retained in its original 12-month format. While seasonal variation in traffic demand—particularly in areas like college campuses—is expected, we refrained from making assumptions about these patterns due to the lack of specific location data in the dataset.

2.3 R Code

```r
#1.1 Data Screening and Variable Selection
df<- df %>% filter(trafficway_type == 'FOUR WAY' |
                     trafficway_type == 'T-INTERSECTION' |
                     trafficway_type == 'L-INTERSECTION' |
                     trafficway_type == 'Y-INTERSECTION' |
                     trafficway_type == 'ROUNDABOUT')

df <- df[c('crash_hour','crash_day_of_week','crash_month','traffic_control_device',
           'weather_condition','lighting_condition','first_crash_type','alignment',
           'roadway_surface_cond','road_defect','most_severe_injury')]

df[] <- lapply(df, function(x) {
  if (is.character(x)) factor(x) else x
})


#1.2 Da                    is.factor(x)
df_sum <- lapply(df, function(x) {
  if (is.factor(x)) {
    list(summary = summary(x), levels = levels(x))
  } else {
    summary(x)
  }
})


min_ratio <- 0.01  # 1% threshold
df_clean <- df
total_n <- nrow(df_clean)

# Identify target column (assumed to be the last column)
target_col <- names(df_clean)[ncol(df_clean)]

# Get all factor/character columns, excluding the target column
qual_cols <- names(Filter(function(x) is.factor(x) || is.character(x), df_clean))
qual_cols <- setdiff(qual_cols, target_col)

# Ensure all relevant columns are factors
df_clean[qual_cols] <- lapply(df_clean[qual_cols], function(col) {
  if (is.character(col)) factor(col) else col
})

# Identify low-frequency levels to filter out
bad_level_map <- lapply(df_clean[qual_cols], function(col) {
  level_counts <- table(col)
  threshold <- ceiling(min_ratio * total_n)
  bad_levels <- names(level_counts[level_counts < threshold])
  bad_levels <- union(bad_levels, "UNKNOWN")  # Also drop "UNKNOWN"
  return(bad_levels)
})
```

```r
# Filter out rows with low-frequency levels
for (col_name in names(bad_level_map)) {
  bad_levels <- bad_level_map[[col_name]]
  df_clean <- df_clean[!df_clean[[col_name]] %in% bad_levels, ]
}

# Filter out rows with target level "REPORTED, NOT EVIDENT"
df_clean <- df_clean[df_clean[[target_col]] != "REPORTED, NOT EVIDENT", ]

# Drop unused factor levels
df_clean[qual_cols] <- lapply(df_clean[qual_cols], droplevels)
df_clean[[target_col]] <- droplevels(df_clean[[target_col]])

# Summary
df_clean_sum <- lapply(df_clean, function(x) {
  if (is.factor(x)) {
    list(summary = summary(x), levels = levels(x))
  } else {
    summary(x)
  }
})

#---Data Transformation---

df_clean$crash_hour <- with(df_clean, ifelse(crash_hour >= 7 & crash_hour <= 9, "AM Peak",
                                ifelse(crash_hour >= 16 & crash_hour <= 18, "PM Peak",
                                       ifelse(crash_hour >= 6 & crash_hour <= 21, "Off-Peak", "Night"))))
df_clean$crash_hour <- factor(df_clean$crash_hour, levels = c("AM Peak", "PM Peak", "Off-Peak", "Night"))

df_clean$crash_day_of_week <- with(df_clean, ifelse(crash_day_of_week >= 1 & crash_day_of_week <= 5, "Weekday", "Weekend"))
df_clean$crash_day_of_week <- factor(df_clean$crash_day_of_week, levels = c('Weekday','Weekend'))

df_clean <- df_clean[, !(names(df_clean) %in% c("alignment", "road_defect"))]
write.csv(df_clean, file ='./Cleaned Traffic Accident Data.csv',row.names = FALSE)
```

3. Split data to train and test set

The dataset was split into training and test sets using an 80/20 proportion. Specifically, 80% of the cleaned dataset was randomly selected as the training set, while the remaining 20% was reserved for testing. The caret package was used to ensure a stratified split based on the target variable. The corresponding R code is shown below:

```r
#2. Split data to train and test set
set.seed(614)

train_index <- createDataPartition(df_clean$most_severe_injury, p = 0.8, list = FALSE)

df_train <- df_clean[train_index,]
df_test <-df_clean[-train_index,]
```

4. Training Models

4.1 K-NN Model

4.1.1 Description

The K-Nearest Neighbors (KNN) model used in this project is a non-parametric classification algorithm that assigns class labels based on the majority class among the nearest training examples in feature space. The model includes one key tuning parameter, k, which defines the number of nearest neighbors to consider when making predictions. Selecting an optimal k is essential for balancing bias and variance — small values of k can lead to overfitting, while large values may oversmooth decision boundaries. In this analysis, a range of odd k values between 1 and 51 was evaluated using 10-fold cross-validation to determine the best-performing configuration.

While KNN is straightforward and intuitive, it is not well-suited for inference since it does not produce coefficients or provide measures of variable significance. The model also does not perform variable selection, as it uses all features for distance calculations unless specified otherwise.

Because KNN relies on distance metrics, it is sensitive to the scale of predictors. Therefore, standardization (centering and scaling) of variables was applied as a preprocessing step to ensure that all features contribute equally to the distance computation.

4.1.2 Tuning Parameters

The KNN model was tuned using 10-fold cross-validation across a grid of odd values for the number of neighbors k, ranging from 1 to 49. The goal was to identify the value of k that yielded the highest classification accuracy. The results showed that accuracy increased steadily with k, peaking at k = 29, where the model achieved a mean accuracy of 77.27% with a Kappa of 0.240. This indicates a moderate agreement between predicted and actual classes, accounting for class imbalance.

Accuracy remained relatively stable for values of k between 17 and 45, with only minor fluctuations in performance. This plateau suggests that the model is not overly sensitive to the exact choice of k within this range, which is a favorable property in terms of robustness. The chosen tuning strategy ensured that the selected model provides a good balance between generalization and complexity.

Table 4-1- K Parameter Tuning Result

k	Accuracy	Kappa	Accuracy SD	Kappa SD
1	0.73542	0.18498	0.00457	0.01489
5	0.76005	0.22127	0.00374	0.01748
9	0.76745	0.23294	0.00355	0.01712
13	0.77100	0.23820	0.00285	0.01459
17	0.77261	0.24131	0.00314	0.01556
21	0.77248	0.24011	0.00303	0.01523
25	0.77273	0.24010	0.00294	0.01495
29	0.77273	0.24035	0.00292	0.01504
33	0.77236	0.23658	0.00324	0.01518
37	0.77217	0.23519	0.00296	0.01418
41	0.77201	0.23360	0.00295	0.01415
45	0.77236	0.23329	0.00317	0.01461
49	0.77204	0.23194	0.00304	0.01366

4.1.3 Training Results Interpretation

The KNN model achieved a peak training accuracy of approximately 77.27% with the optimal number of neighbors determined to be k = 29 through 10-fold cross-validation. While this accuracy appears

promising on the surface, it is essential to evaluate it considering the null accuracy, which represents the accuracy of a naïve model that predicts only the most frequent class.

In this dataset, the class 'NO INDICATION OF INJURY' constitutes approximately 74% of all training samples. This high proportion introduces a significant sample imbalance, which creates a strong bias in favor of the majority class. As a result, even a trivial model that always predicts 'NO INDICATION OF INJURY' would achieve a baseline accuracy close to 74%. The KNN model's performance, though slightly better than this null accuracy, suggests that its predictive improvement is marginal and potentially driven more by class prevalence than by its ability to accurately distinguish among all injury severities.

This bias is further reflected in the relatively low Kappa statistic (≈0.240), which adjusts for chance agreement and indicates limited success in capturing minority class patterns. The results highlight an important early observation in the modeling process: the imbalanced nature of the response variable must be carefully accounted for, as it can skew model performance metrics and hinder a model's ability to generalize to less frequent but critical outcomes like 'FATAL' or 'INCAPACITATING INJURY.'

Table 4-2 – Sample Size of Different Injury Severity

FATAL	INCAPACITATING INJURY	NONCAPACITATING INJURY	NO INDICATION OF INJURY	Total
87	1663	29401	8575	39726

4.1.4 R code

```r
#3. Training Model
# --- 3.1 --- KNN ---
k_grid <- expand.grid(k = seq(1, 51, by = 4))

ctrl <- trainControl(method = "cv", number = 10)

knn_model <- train(
  most_severe_injury ~ .,
  data = df_train,
  method = "knn",
  trControl = ctrl,
  preProcess = c("center", "scale"),
  tuneGrid = k_grid
)

knn_model$results #show results
knn_model$bestTune #best tune
```

4.2 LASSO

4.2.1 Description

LASSO is a parametric model, meaning it makes specific assumptions about the functional form of the relationship between predictors and the response variable. LASSO regression includes a key tuning

parameter, λ (lambda), which controls the strength of the L1 regularization penalty applied to the model's coefficients. Larger values of lambda increase the penalty, forcing more coefficients toward zero, thereby simplifying the model. In this study, lambda was selected using 10-fold cross-validation to balance model complexity with classification accuracy. LASSO can also be used for inference. The non-zero coefficients selected by the model provide insight into which variables are most relevant to predicting the response. For example, in this study, the model identified specific crash types (e.g., pedestrian and cyclist involvement) as being associated with certain injury severity. An important advantage of LASSO is its ability to perform automatic variable selection. By shrinking some coefficients exactly to zero, it effectively excludes non-informative predictors from the model, enhancing interpretability and reducing overfitting — especially valuable when dealing with high-dimensional or sparse data. Lastly, LASSO requires that predictors be standardized (i.e., centered and scaled), because the penalty term depends on the magnitude of the coefficients. Without standardization, variables with larger scales would be penalized more heavily, skewing the model. In this analysis, all numeric features were standardized prior to model fitting to ensure fair regularization.

4.2.2 Tuning Parameters

The LASSO multinomial logistic regression model includes a critical tuning parameter, lambda (λ), which controls the strength of the L1 regularization applied to the model's coefficients. A higher lambda value increases the penalty on the absolute size of the coefficients, effectively shrinking less informative coefficients toward zero. This property allows the model to perform variable selection by excluding unimportant predictors.

In this analysis, a grid of lambda values ranging from 100 to 0.0001 was defined on a logarithmic scale. The optimal lambda value was selected using 10-fold cross-validation, which helps to estimate out-of-sample error and prevent overfitting. The best-performing lambda, identified by minimizing the cross-validated classification error, was $\lambda = 0.0305$, a relatively mild regularization strength. This level allowed the model to retain only a small subset of features, improving generalizability while maintaining interpretability.

4.2.3 Training Results Interpretation

The fitted LASSO model selected a minimal set of predictors, with only a few features having non-zero coefficients—most notably under the 'NO INDICATION OF INJURY' class. These retained predictors included crash types such as 'REAR END', 'SIDESWIPE SAME DIRECTION', 'PEDESTRIAN', and 'PEDALCYCLIST', highlighting that the type of first crash plays a meaningful role in predicting the severity of injury. For instance, crashes involving pedestrians or cyclists were negatively associated with the 'NO INDICATION OF INJURY' class, suggesting these crash types are more likely to result in an injury, which aligns with traffic safety intuition.

However, for other injury levels (e.g., 'FATAL', 'NONCAPACITATING INJURY' and 'INCAPACITATING INJURY'), all predictors were shrunk to zero. This is likely due to severe class

imbalance, shown in Table 4-2, with majority of the samples falling into the 'NO INDICATION OF INJURY' category. As a result, the model favored features that helped differentiate this dominant class, while regularizing away features that offered limited predictive power for the underrepresented categories. This suggests that while the model has utility in identifying conditions related to low-risk outcomes, further steps such as class rebalancing may be necessary to improve prediction of more severe injuries.

Table 4-3- Coefficients for the LASSO Model

Coefficients	FATAL	INCAPACITATING INJURY	NONCAPACITATING INJURY	NO INDICATION OF INJURY
Intercept	-3.368	-0.423	2.574	1.217
crash_hourPM Peak	-	-	-	-
crash_hourOff-Peak	-	-	-	-
crash_hourNight	-	-	-	-
crash_day_of_weekWeekend	-	-	-	-
crash_month	-	-	-	-
traffic_control_deviceSTOP SIGN/FLASHER	-	-	-	-
traffic_control_deviceTRAFFIC SIGNAL	-	-	-	-
weather_conditionCLOUDY/OVERCAST	-	-	-	-
weather_conditionRAIN	-	-	-	-
weather_conditionSNOW	-	-	-	-
lighting_conditionDARKNESS, LIGHTED ROAD	-	-	-	-
lighting_conditionDAWN	-	-	-	-
lighting_conditionDAYLIGHT	-	-	-	-
lighting_conditionDUSK	-	-	-	-
first_crash_typeFIXED OBJECT	-	-	-	-
first_crash_typePARKED MOTOR VEHICLE	-	-	-	-
first_crash_typePEDALCYCLIST	-	-	-1.500	-
first_crash_typePEDESTRIAN	-	-	-2.247	-
first_crash_typeREAR END	-	-	0.167	-
first_crash_typeSIDESWIPE SAME DIRECTION	-	-	0.294	-
first_crash_typeTURNING	-	-	0.017	-
roadway_surface_condSNOW OR SLUSH	-	-	-	-
roadway_surface_condWET	-	-	-	-

4.2.4 R Code

```r
# ---3.2 --- LASSO
set.seed(614)

x_train <- model.matrix(~ ., data = df_train[, -ncol(df_train)])[ , -1]
y_train <- df_train$most_severe_injury

grid <- 10^seq(2, -4, length = 100)  # From 100 to 0.0001

lasso_model <- cv.glmnet(
  x = x_train,
  y = y_train,
  family = "multinomial",
  alpha = 1,
  lambda = grid,
  type.measure = "class",
  nfolds = 10
)

plot(lasso_model)
lasso_model$lambda.min
coef(lasso_model, s = lasso_model$lambda.min)
lasso_train_accuracy <- 1 - lasso_model$cvm[lasso_model$lambda == lasso_model$lambda.min] #training error
```

4.3 Multinomial GAM

4.3.1 Description

Since the textbook example only covers binary classification using Generalized Additive Models (GAMs), the multinomial logistic GAM implementation in this project was developed based on supplementary research and may contain minor approximations. The model was fitted using the vglm() function from the VGAM package to predict the multiclass injury severity outcome. The GAM model developed in this project is a parametric models, combining the parametric structure of generalized linear models with the a non-linear smooth function: the crash_month variable was modeled using a spline, allowing for the potential capture of seasonal or cyclical trends in crash severity.

The inclusion of a spline introduces a tuning parameter—the degrees of freedom (df)—which controls the smoothness of the estimated function. Higher degrees of freedom allow the spline to capture more detailed variation in the data, while lower values enforce greater smoothness and help prevent overfitting. In this analysis, a range of degrees of freedom was tested manually to evaluate model flexibility and fit.

The multinomial GAM can be used for inference, particularly in identifying which predictors are significantly associated with changes in injury severity. The model output includes coefficient estimates, standard errors, and z-values for each term, allowing interpretation of both the direction and magnitude of predictor effects. However, as with any additive model, caution should be exercised when interpreting coefficients from smooth terms or when separation occurs in rare outcome classes.

Unlike regularized models such as LASSO, this implementation of GAM does not include automatic variable selection. All predictors included in the model specification remain active unless explicitly removed during model development. Finally, standardization of predictors is not required when using vglm(), as the function internally handles both categorical predictors and numeric splines appropriately without the need for preprocessing.

4.3.2 Tuning Parameters

To tune the flexibility of the spline applied to the crash_month variable, a grid search was conducted across degrees of freedom (df) ranging from 4 to 10 using 10-fold cross-validation on the training dataset. The average classification accuracy was computed for each df setting. Results indicated that all df values produced identical cross-validated accuracy across folds (I am really not sure if that make sense), suggesting that the spline flexibility had no impact on model performance. Inspection of the model coefficients confirmed this, as the spline terms for crash_month were not statistically significant (p-values > 0.05), indicating that this variable did not meaningfully contribute to prediction. Therefore, the final model retained df = 4 for simplicity and interpretability.

Table 4-4 Degree of Freedom Tuning Result

No.	df	Accuracy
1	4	0.774
2	5	0.774
3	6	0.774
4	7	0.774
5	8	0.774
6	9	0.774
7	10	0.774

4.3.3 Training Results Interpretation

The multinomial logistic GAM model estimates the log-odds of each injury severity level relative to the reference category, 'NO INDICATION OF INJURY'. In the model output, the predicted categories are indexed as follows:

Category 1 corresponds to 'FATAL' injuries,

Category 2 corresponds to 'INCAPACITATING INJURY',

Category 3 corresponds to 'NONINCAPACITATING INJURY'.

Thus, the model outputs three sets of coefficients, each representing the influence of predictors on the log-odds of being in one of these three injury severity categories versus having no indication of injury. The results, shown in While the model results offer useful insights, it is important to recognize the limitations introduced by the imbalanced distribution of injury severity categories. In particular, some variables—such as weather conditions like snow or rain—were found to be statistically significant only for incapacitating injuries (Category 2), but not for fatal or non-incapacitating injuries. This does not necessarily imply that these factors have no effect on the other injury types. Rather, the observed significance may be influenced by the underrepresentation of fatal cases and the dominance of non-injury records in the dataset. These imbalances introduce challenges in estimating effects with stability and precision. As such, the findings should be interpreted with caution, and future studies would benefit from incorporating a larger and more balanced dataset to more accurately assess the influence of these predictors across all injury severity levels.

Table 4-5, indicate that several predictors are statistically significant and contribute meaningfully to the classification of injury severity.

Among the most informative variables, first_crash_type shows strong and consistent effects. Crashes involving pedestrians or pedalcyclists are associated with substantially higher odds of fatal, incapacitating, and non-incapacitating injuries (all $p < 0.001$), highlighting the vulnerability of these road users. In contrast, crashes categorized as rear-end, turning, or sideswipe are significantly associated with lower injury severity ($p < 0.001$), suggesting they are typically less dangerous scenarios.

Time-of-day also plays a role: nighttime crashes significantly increase the odds of both incapacitating and non-incapacitating injuries compared to peak-hour crashes ($p = 0.003$ and $p < 0.001$, respectively), likely reflecting reduced visibility or increased speeding during nighttime. However, other time categories like AM and PM peak hours were not significant, indicating limited variability during daytime periods.

Weather conditions also influence injury severity. Rain and especially snow are associated with significantly lower odds of incapacitating injuries ($p = 0.004$ and $p < 0.001$, respectively), possibly due to more cautious driving behaviors under adverse conditions. Meanwhile, lighting condition and day of week did not show strong statistical significance in most categories, suggesting more marginal effects after controlling for other variables.

Notably, the spline terms for crash_month were not statistically significant, confirming that there is no strong seasonal pattern in injury severity once other predictors are included. This also aligns with prior tuning results, which showed no improvement in classification accuracy from increasing the spline flexibility for this variable.

While the model results offer useful insights, it is important to recognize the limitations introduced by the imbalanced distribution of injury severity categories. In particular, some variables—such as weather conditions like snow or rain—were found to be statistically significant only for incapacitating injuries (Category 2), but not for fatal or non-incapacitating injuries. This does not necessarily imply that these factors have no effect on the other injury types. Rather, the observed significance may be influenced by the underrepresentation of fatal cases and the dominance of non-injury records in the dataset. These imbalances introduce challenges in estimating effects with stability and precision. As such, the findings should be interpreted with caution, and future studies would benefit from incorporating a larger and more balanced dataset to more accurately assess the influence of these predictors across all injury severity levels.

Table 4-5 – Coefficients Summary of Multinomial Logistic GAM Model

Coefficients:	**Estimate**	**Std. Error**	**z value**	**Pr(>|z|)**	
(Intercept):1	-7.752792	1.31141	-5.912	3.38E-09	***
(Intercept):2	-2.597849	0.222936	-11.653	< 2e-16	***
(Intercept):3	-1.309149	0.119679	-10.939	< 2e-16	***
crash_hourPM Peak:1	-0.018674	0.58437	-0.032	0.974507	
crash_hourPM Peak:2	0.016869	0.110315	0.153	0.878462	
crash_hourPM Peak:3	-0.016191	0.052908	-0.306	0.759581	
crash_hourOff-Peak:1	0.462442	0.496891	0.931	0.352024	

crash_hourOff-Peak:2	0.125947	0.094857	1.328	0.184258	
crash_hourOff-Peak:3	-0.007861	0.04533	-0.173	0.862328	
crash_hourNight:1	1.008587	0.59126	1.706	0.08804	.
crash_hourNight:2	0.38653	0.130486	2.962	0.003054	**
crash_hourNight:3	0.342749	0.065531	5.23	1.69E-07	***
s(crash_month, df = 4):1	0.006228	0.035274	0.177	0.859846	
s(crash_month, df = 4):2	-0.007698	0.008892	-0.866	0.386692	
s(crash_month, df = 4):3	0.006823	0.004507	1.514	0.129993	
traffic_control_deviceSTOP SIGN/FLASHER:1	0.448872	0.637604	0.704	0.481434	
traffic_control_deviceSTOP SIGN/FLASHER:2	-0.441683	0.114055	-3.873	0.000108	***
traffic_control_deviceSTOP SIGN/FLASHER:3	-0.072705	0.059224	-1.228	0.219583	
traffic_control_deviceTRAFFIC SIGNAL:1	1.011752	0.606274	1.669	0.095156	.
traffic_control_deviceTRAFFIC SIGNAL:2	0.117661	0.10292	1.143	0.252945	
traffic_control_deviceTRAFFIC SIGNAL:3	0.198822	0.055346	3.592	0.000328	***
lighting_conditionDARKNESS, LIGHTED ROAD:1	1.289701	1.019333	1.265	0.205785	
lighting_conditionDARKNESS, LIGHTED ROAD:2	0.016984	0.168485	0.101	0.919705	
lighting_conditionDARKNESS, LIGHTED ROAD:3	0.128461	0.094076	1.365	0.172097	
lighting_conditionDAWN:1	0.362271	1.42058	0.255	0.798711	
lighting_conditionDAWN:2	0.169834	0.239172	0.71	0.477646	
lighting_conditionDAWN:3	0.051445	0.139218	0.37	0.711736	
lighting_conditionDAYLIGHT:1	0.548704	1.034822	0.53	0.595946	
lighting_conditionDAYLIGHT:2	-0.130053	0.168024	-0.774	0.438924	
lighting_conditionDAYLIGHT:3	0.178825	0.09382	1.906	0.056643	.
lighting_conditionDUSK:1	1.687479	1.130541	1.493	0.135534	
lighting_conditionDUSK:2	-0.094174	0.238473	-0.395	0.692913	
lighting_conditionDUSK:3	0.121993	0.126225	0.966	0.333809	
weather_conditionCLOUDY/OVERCAST:1	0.697783	0.526701	1.325	0.185232	
weather_conditionCLOUDY/OVERCAST:2	-0.338707	0.180333	-1.878	0.06035	.
weather_conditionCLOUDY/OVERCAST:3	-0.101	0.080524	-1.254	0.209738	
weather_conditionRAIN:1	-0.082916	0.384208	-0.216	0.829136	
weather_conditionRAIN:2	-0.29648	0.103249	-2.872	0.004085	**
weather_conditionRAIN:3	-0.044806	0.049199	-0.911	0.36245	
weather_conditionSNOW:1	-0.138644	0.732249	-0.189	0.849827	
weather_conditionSNOW:2	-0.898483	0.249122	-3.607	0.00031	***
weather_conditionSNOW:3	-0.172827	0.097894	-1.765	0.077488	.
first_crash_typeFIXED OBJECT:1	-0.302502	0.739446	-0.409	0.682472	
first_crash_typeFIXED OBJECT:2	-0.414945	0.201171	-2.063	0.039147	*
first_crash_typeFIXED OBJECT:3	-0.451514	0.098818	-4.569	4.90E-06	***
first_crash_typePARKED MOTOR VEHICLE:1	0.832028	0.620568	1.341	0.180001	
first_crash_typePARKED MOTOR VEHICLE:2	-1.548035	0.415127	-3.729	0.000192	***
first_crash_typePARKED MOTOR VEHICLE:3	-1.436209	0.175788	-8.17	3.08E-16	***
first_crash_typePEDALCYCLIST:1	1.208897	0.739355	1.635	0.102035	
first_crash_typePEDALCYCLIST:2	2.135519	0.125917	16.96	< 2e-16	***
first_crash_typePEDALCYCLIST:3	2.000841	0.084327	23.727	< 2e-16	***
first_crash_typePEDESTRIAN:1	3.629445	0.331998	10.932	< 2e-16	***
first_crash_typePEDESTRIAN:2	3.357932	0.114268	29.386	< 2e-16	***
first_crash_typePEDESTRIAN:3	2.836081	0.093057	30.477	< 2e-16	***
first_crash_typeREAR END:1	-14.983164	294.445567	NA	NA	
first_crash_typeREAR END:2	-1.380284	0.134128	-10.291	< 2e-16	***
first_crash_typeREAR END:3	-1.119972	0.056096	-19.965	< 2e-16	***
first_crash_typeSIDESWIPE SAME DIRECTION:1	-1.835529	1.02271	-1.795	0.07269	.
first_crash_typeSIDESWIPE SAME DIRECTION:2	-2.415183	0.294273	-8.207	2.26E-16	***
first_crash_typeSIDESWIPE SAME DIRECTION:3	-1.812443	0.100833	-17.975	< 2e-16	***
first_crash_typeTURNING:1	-0.652765	0.315992	-2.066	0.038851	*
first_crash_typeTURNING:2	-0.744884	0.07588	-9.817	< 2e-16	***
first_crash_typeTURNING:3	-0.638616	0.03526	-18.111	< 2e-16	***
crash_day_of_weekWeekend:1	-0.322494	0.275528	-1.17	0.241817	
crash_day_of_weekWeekend:2	0.054003	0.062601	0.863	0.388331	
crash_day_of_weekWeekend:3	-0.077786	0.032196	-2.416	0.015692	*

4.3.4 R code

```r
# --- 3.3 --- Multinomial Log GAM
df_grid <- 4:10
k <- 10  # Number of folds

folds <- createFolds(df_train$most_severe_injury, k = k)

models <- list()
results <- data.frame(df = integer(), Fold = integer(), Accuracy = nu

# Loop over each df
for (df_i in df_grid) {
  cat("Tuning df =", df_i, "\n")

  for (i in seq_along(folds)) {
    cat("  Fold", i, "\n")

    val_idx <- folds[[i]]
    train_fold <- df_train[-val_idx, ]
    val_fold <- df_train[val_idx, ]

    # Fit the GAM model
    model <- vglm(
      most_severe_injury ~
        crash_hour +
        s(crash_month, df = df_i) +
        traffic_control_device +
        lighting_condition +
        weather_condition +
        first_crash_type +
        crash_day_of_week,
      family = multinomial(refLevel = "NO INDICATION OF INJURY"),
      data = train_fold
    )

    # Predict on validation fold
    probs <- predict(model, newdata = val_fold, type = "response")
    preds <- colnames(probs)[apply(probs, 1, which.max)]

    # Accuracy
    acc <- mean(preds == val_fold$most_severe_injury)

    # Save result
    results <- rbind(results, data.frame(df = df_i, Fold = i, Accuracy = acc))
  }
}

GAM_accuracy <- aggregate(Accuracy ~ df, data = results, FUN = mean)

final_gam <- vglm(
  most_severe_injury ~
    crash_hour +
    s(crash_month, df = 4) +
    traffic_control_device +
    lighting_condition +
    weather_condition +
    first_crash_type +
    crash_day_of_week,
  family = multinomial(refLevel = "NO INDICATION OF INJURY"),
  data = df_train
)
summary(final_gam)
```

Show in new window

4.4 Single Tree Model

4.4.1 Description

The decision tree model in this analysis was constructed using the rpart package in R. Decision trees are considered non-parametric models because they do not assume a particular functional form between

predictors and the outcome. Instead, they recursively partition the data space using binary splits, forming a hierarchical structure that is both intuitive and interpretable.

This model includes a key tuning parameter known as the complexity parameter (cp). The cp controls how much the model improves the fit at each split. Specifically, a split is only performed if it reduces the overall lack of fit by a factor greater than the specified cp value. This parameter directly influences the size of the tree and acts as a form of regularization: smaller values of cp allow more complex trees, while larger values encourage simpler, more generalizable models. Tuning cp helps balance the trade-off between underfitting and overfitting.

While decision trees do support interpretable decision rules. Each path from root to leaf can be understood as a logical rule associating predictor conditions with a predicted class, making them highly explainable for decision-making contexts.

Another important property of decision trees is that they perform implicit variable selection. Only variables that meaningfully reduce impurity are included in the final model. Irrelevant or redundant predictors are automatically excluded during tree construction, without requiring additional preprocessing or feature elimination.

Finally, standardization of predictors is not required for decision trees. The model is invariant to monotonic transformations of the input variables and can directly handle both categorical and numerical features without scaling. Categorical predictors are automatically evaluated for optimal groupings of levels during the splitting process.

4.4.2 Tuning Parameters

To select an appropriate cp value, 10-fold cross-validation was performed using the built-in xval option in the rpart package. The cross-validated misclassification error was evaluated across different tree sizes, with the goal of identifying the subtree that minimized this error. Based on the CP results shown in Table 4-6, the optimal value occurred at a tree with only one internal split—the simplest non-trivial tree. Although deeper trees marginally reduced the training error, they did not yield lower cross-validated error, suggesting potential overfitting. Therefore, the final model was pruned to this optimal size.

Table 4-6 – Complexity Parameter Table Result

	CP	nsplit	rel error	xerror	xstd
1	1.32E-01	0	1	1	0.009465
2	1.21E-04	1	0.86769	0.86769	0.0090192
3	1.08E-04	34	0.86333	0.87895	0.0090603
4	1.04E-04	73	0.85837	0.88234	0.0090726
5	1.01E-04	84	0.85716	0.88355	0.009077
6	9.08E-05	90	0.85655	0.8873	0.0090905
7	8.88E-05	115	0.85413	0.88875	0.0090957
8	8.65E-05	134	0.85244	0.8942	0.0091151
9	8.07E-05	141	0.85183	0.89747	0.0091267
10	7.57E-05	162	0.85014	0.89989	0.0091352

11	6.73E-05	185	0.84784	0.90268	0.009145
12	6.60E-05	229	0.84433	0.91248	0.0091792
13	6.05E-05	261	0.84191	0.91284	0.0091805
14	5.65E-05	358	0.83561	0.91442	0.0091859
15	5.19E-05	396	0.83247	0.91648	0.0091931
16	4.84E-05	403	0.8321	0.92204	0.0092122
17	4.40E-05	418	0.83138	0.92301	0.0092155
18	4.27E-05	433	0.83065	0.93221	0.0092467
19	4.04E-05	452	0.8298	0.93221	0.0092467
20	3.46E-05	511	0.82738	0.93584	0.0092589
21	3.03E-05	540	0.82629	0.94056	0.0092747
22	2.69E-05	629	0.82339	0.94165	0.0092784
23	2.42E-05	653	0.82266	0.94553	0.0092912
24	2.02E-05	705	0.82133	0.94819	0.0093
25	1.73E-05	747	0.82048	0.94964	0.0093048
26	1.51E-05	775	0.82	0.95061	0.009308
27	1.01E-05	783	0.81988	0.95085	0.0093088
28	0.00E+00	795	0.81976	0.95085	0.0093088

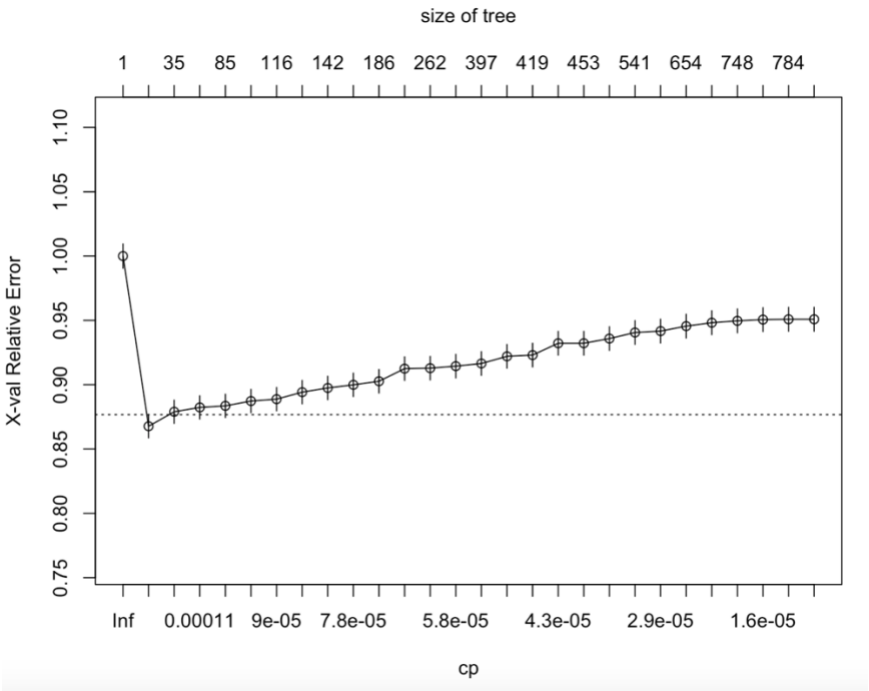

Figure 4-1 – Complexity Parameter Plot

4.4.3 Training Results Interpretation

The final decision tree model was pruned to a single binary split, using only the variable first_crash_type. This outcome emphasizes the strong predictive power of crash behavior type in determining injury severity outcomes. The model grouped the eight levels of first_crash_type into two subsets based on an internal split structure coded as 'LLLRRLLL', indicating which levels were sent to the left or right branch of the tree.

Specifically, the crash types routed to the left node were: ANGLE, FIXED OBJECT, PARKED MOTOR VEHICLE, REAR END, SIDESWIPE SAME DIRECTION, and TURNING. The right node included only two crash types: PEDALCYCLIST and PEDESTRIAN.

The left node was predicted as 'NO INDICATION OF INJURY'. This node contained crash types that are more common and typically associated with lower-severity or property-damage-only incidents. The class distribution within this group included 78.4% nonincapacitating injuries, 18.3% no injury, and only a small proportion of more serious outcomes. Despite nonincapacitating injury being the largest proportion, the model selected 'NO INDICATION OF INJURY' as the predicted class due to the impurity reduction criterion used in the tree algorithm.

The right node was predicted as 'NONINCAPACITATING INJURY' and included crash types involving more vulnerable road users—pedestrians and pedal cyclists. Although this node still included a high proportion of no-injury cases (65.3%), it also contained a meaningful share of incapacitating and nonincapacitating injuries. The tree assigned 'NONINCAPACITATING INJURY' as the predicted class for this node, reflecting the elevated injury risk associated with these crash types.

These findings reinforce the critical role of crash behavior type in injury severity classification. The model effectively used first_crash_type to segment the data into high- and low-risk crash profiles. However, the model did not predict the 'FATAL' or 'INCAPACITATING INJURY' classes in either terminal node. This likely reflects the severe imbalance in the response variable, where such outcomes are underrepresented. While the simplicity of the model supports strong interpretability and practical application, future modeling efforts may benefit from class reweighting or alternative algorithms to better capture rare but high-severity outcomes.

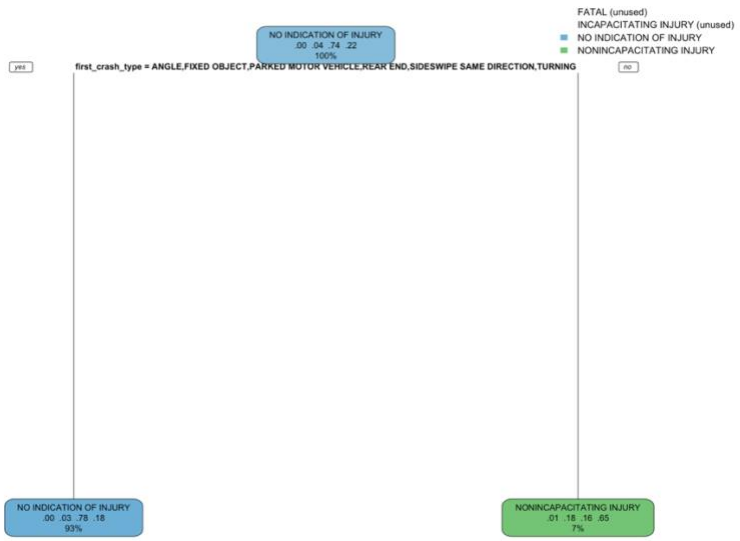

Figure 4-2Single Tree Model Plot

4.4.4 R code

```r
# --- 3.4 --- Tree Model

tree_model <- rpart(
  most_severe_injury ~ .,
  data = df_train,
  method = "class",
  control = rpart.control(
    xval = 10,
    minbucket = 2,
    cp = 0
  )
)

printcp(tree_model)
plotcp(tree_model)
best_cp<- tree_model$cptable[which.min(tree_model$cptable[,"xerror"]), "CP"]

tree_pruned <- prune(tree_model, cp = best_cp)

rpart.plot(tree_pruned, type = 2, extra = 104, cex = 0.7)
summary(tree_pruned)
```

4.5 Random Forest

4.5.1 Description

The Random Forest is a non-parametric ensemble model that constructs a collection of decision trees using bootstrap samples and random feature selection at each split. This structure allows the model to capture complex interactions and nonlinear relationships without assuming a specific functional form, making it especially useful for high-dimensional or heterogeneous data.

Random Forests involve several tuning parameters, the most important being mtry, which controls the number of predictors randomly sampled for consideration at each node split. Additional parameters include ntree (the total number of trees) and nodesize (minimum observations per leaf node). In this analysis, mtry was tuned using out-of-bag (OOB) resampling, and ntree was set to 500.

Although Random Forests do not provide traditional coefficient estimates or hypothesis tests, they offer an interpretable output in the form of variable importance metrics. These metrics assess how much each predictor contributes to the model's accuracy and can be used for inference about feature relevance, particularly when comparing the relative influence of different variables.

The model does not perform automatic variable selection in a strict sense; however, its design naturally downweights uninformative predictors, as variables that do not improve decision splits are used infrequently across the ensemble. Finally, Random Forests do not require predictor standardization, since the tree-splitting mechanism is insensitive to the scale of input features.

4.5.2 Tuning Parameters

The Random Forest model was trained using the full set of 31,782 crash records and eight selected predictors to classify injury severity into four categories. The model was tuned across a range of mtry values — the number of variables randomly selected at each tree split — from 1 to 8. The highest training

accuracy was achieved when mtry = 3, with an overall accuracy of 77.44% and a corresponding Kappa of 0.239. This suggests moderate agreement beyond chance and indicates the model is learning meaningful structure in the data, though some misclassification likely remains due to class imbalance.

Table 4-7 - Resampling results across tuning parameters

mtry	Accuracy	Kappa
1	0.740073	0
2	0.7729532	0.2280041
3	0.7744321	0.2391768
4	0.7742433	0.2384551
5	0.7742118	0.2382216
6	0.7737713	0.2367768
7	0.7726701	0.2335859
8	0.7713171	0.2298799

4.5.3 Training Results Interpretation

The final Random Forest model, trained with mtry = 3, provided interpretable insight into the relative influence of each predictor on injury severity classification. Variable importance was assessed using the Mean Decrease in Accuracy metric, where higher values indicate a greater impact on the model's predictive performance.

The most important variable by a large margin was first_crash_typePEDESTRIAN, with an importance score of 100, suggesting that crashes involving pedestrians are strongly associated with more severe injury outcomes. The second most influential predictor was first_crash_typePEDALCYCLIST (45.0), followed by REAR END (17.2), TURNING (16.3), and SIDESWIPE SAME DIRECTION (14.9). These results indicate that the type of crash—particularly vulnerable road users and certain impact scenarios—plays a dominant role in determining the level of injury severity.

Among the temporal and environmental variables, crash_month had moderate influence (13.0), potentially capturing seasonal trends in crash frequency or severity. Contextual factors such as traffic_control_deviceTRAFFIC SIGNAL (3.1), crash_hourNight (2.3), and crash_day_of_weekWeekend (2.2) also contributed modestly, indicating that both timing and roadway control conditions affect crash outcomes.

Other predictors like lighting_conditionDARKNESS, LIGHTED ROAD, weather_conditionRAIN, and roadway_surface_condWET had relatively lower importance (all < 2), suggesting more limited roles in the model's classification ability. However, their presence in the model still supports their potential value in understanding the crash environment holistically.

In summary, the Random Forest model emphasized crash behavior, especially those involving vulnerable users, as the most predictive feature group. This aligns well with real-world expectations and highlights critical areas for injury prevention interventions.

Table 4-8 – Variable Importance Table

Variable	Overall
first_crash_typePEDESTRIAN	100
first_crash_typePEDALCYCLIST	45.034
first_crash_typeREAR END	17.1578
first_crash_typeTURNING	16.2572
first_crash_typeSIDESWIPE SAME DIRECTION	14.8657
crash_month	12.9996
traffic_control_deviceTRAFFIC SIGNAL	3.0921
crash_hourNight	2.2603
crash_day_of_weekWeekend	2.1701
traffic_control_deviceSTOP SIGN/FLASHER	1.9541
first_crash_typePARKED MOTOR VEHICLE	1.8175
roadway_surface_condWET	1.5661
lighting_conditionDARKNESS, LIGHTED ROAD	1.4186
lighting_conditionDAYLIGHT	1.3679
crash_hourOff-Peak	1.2953
crash_hourPM Peak	1.2895
weather_conditionRAIN	1.0638
weather_conditionCLOUDY/OVERCAST	0.7107
first_crash_typeFIXED OBJECT	0.5976
weather_conditionSNOW	0.3715

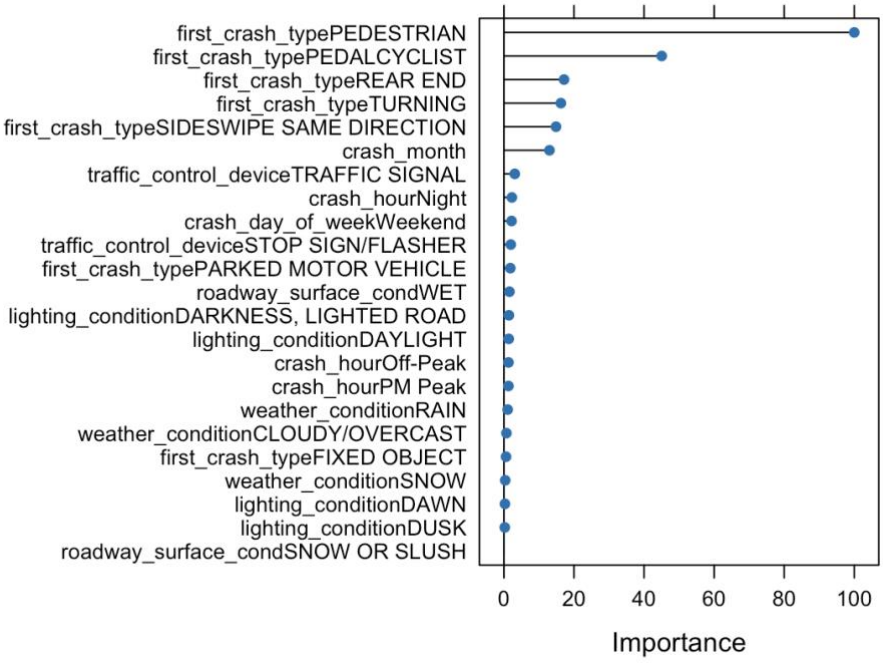

Figure 4-3 – Variable Importance Plot

4.5.4 R code

```r
# --- 3.5 --- Random Forest
ctrl <- trainControl(method = "oob")

mtry_grid <- expand.grid(mtry = 1:(ncol(df_train) - 1))  # exclude target

rf_model <- train(
  most_severe_injury ~ .,
  data = df_train,
  method = "rf",
  trControl = ctrl,
  tuneGrid = mtry_grid,
  ntree = 500
)

print(rf_model)
varImp(rf_model)
plot(varImp(rf_model), top = 23)
```

4.6 Support Vector Machines

4.6.1 Description

The Support Vector Machine (SVM) model applied in this analysis is a non-parametric classification model using a radial basis function (RBF) kernel, implemented via the svm() function from the e1071 package. This model includes two key tuning parameters: the cost parameter (C), which controls the trade-off between classification accuracy and margin width, and gamma, which defines the influence range of each support vector in the feature space. For this model, gamma was set to 1 and cost to 0.2.

Although SVM models do not provide interpretable coefficients like logistic regression, they do enable a form of inference by analyzing the distribution and number of support vectors, as well as evaluating performance across classes. However, they are not ideal for interpreting the direction or magnitude of individual variable effects.

SVM does not perform variable selection automatically — all predictors are used in the classification unless filtered prior to training. Additionally, because SVM relies on distance-based kernel functions, it is strongly recommended to standardize predictors, especially when numeric features are on different scales. In this project, categorical variables were used without standardization, which is generally acceptable when using kernel SVMs with factor inputs.

4.6.2 Tuning Parameters

The Support Vector Machine (SVM) model was configured with a radial basis function (RBF) kernel and manually assigned values for the key tuning parameters: cost and gamma. The cost parameter, which controls the trade-off between margin width and classification error, was set to 0.2, while gamma, which influences the complexity of the decision boundary, was fixed at 1. Ideally, a grid search or cross-validation procedure would be used to systematically evaluate a range of values and identify the optimal

combination for these parameters. However, due to computational constraints and long model training times, full hyperparameter tuning was not conducted. As a result, the selected cost value may not reflect the best-performing configuration, and further exploration could potentially improve model accuracy and generalizability — especially for underrepresented classes in the dataset.

4.6.3 Training Results Interpretation

The Support Vector Machine (SVM) model using a radial basis function kernel was evaluated on the training dataset across four injury severity classes. Table 4.9 presents the confusion matrix-derived metrics for each category. The model demonstrated strong performance in predicting the most prevalent class, 'NO INDICATION OF INJURY,' which comprises approximately 74% of all training observations. This class proportion defines the null accuracy — the expected performance of a model that always predicts the majority class without learning any feature relationships.

In this case, the SVM model achieved an overall training accuracy of approximately 77%, only marginally outperforming the null accuracy. While this slight increase may appear favorable, it is important to note that most of this gain is concentrated in the dominant class. The SVM attained a sensitivity of 98.9% and precision of 77.7% for the 'NO INDICATION OF INJURY' class, resulting in a strong F1 score of 0.87.

In contrast, the model struggled with the other three classes, particularly the minority categories. For 'NONINCAPACITATING INJURY,' the model showed a sensitivity of only 18.3%, meaning that the vast majority of actual cases were missed. While the precision was relatively strong at 67.5%, the low recall led to an F1 score of just 0.29, indicating poor overall performance for this category.

Most notably, the SVM failed to identify any instances of the 'FATAL' and 'INCAPACITATING INJURY' classes. Both had a sensitivity of 0.00, and since the model never predicted these classes, their precision and F1 scores were undefined. These results suggest that the model entirely ignored the minority classes during prediction, despite perfect specificity (1.00), which simply reflects that it correctly labeled other observations as 'not fatal' or 'not incapacitating.'

This outcome highlights a common limitation of standard SVM models when applied to imbalanced datasets. While the model achieved high overall accuracy, this performance was heavily skewed by its success in identifying the majority class. It failed to capture the rare but critically important injury outcomes — particularly fatal and incapacitating injuries — which significantly reduces its practical utility in safety-critical applications. These results underscore the need for class-balancing strategies or weighted learning approaches in future modeling efforts.

Table 4-9 – Confusion Matrix Table

	FATAL	**INCAPACITATING INJURY**	**NO INDICATION OF INJURY**	**NONINCAPACITATING INJURY**
Sensitivity	0.000	0.000	0.989	0.183

Specificity	1.000	1.000	0.192	0.976
Pos Pred Value	NaN	NaN	0.777	0.675
Neg Pred Value	0.998	0.958	0.855	0.813
Precision	NA	NA	0.777	0.675
Recall	0.000	0.000	0.989	0.183
F1	NA	NA	0.870	0.288
Prevalence	0.002	0.042	0.740	0.216
Detection Rate	0.000	0.000	0.732	0.039
Detection Prevalence	0.000	0.000	0.942	0.584
Balanced Accuracy	0.500	0.500	0.590	0.579

4.6.4 R code

```r
# --- 3.6 --- SVM
svm_model <- svm(
  most_severe_injury ~ .,
  data = df_train,
  type = "C-classification",
  kernel = "radial",          # or "linear", "polynomial", etc.
  gamma = 1,
  cost = 0.2
)
summary(svm_model)
svm_pred <- predict(svm_model, df_train)
conf_mat <- confusionMatrix(svm_pred, df_train$most_severe_injury)
conf_mat$overall["Accuracy"]
conf_mat$byClass
```

5. Testing Models

5.1 Testing Results

5.1.1 General Performance

Table 5-1presents the overall accuracy of each classification model evaluated on the test dataset. Accuracy here refers to the proportion of correct predictions across all severity classes. The results indicate comparable performance among most models, with four of them — LASSO (Multinomial Logistic Regression), Generalized Additive Model (GAM), pruned Decision Tree, and Random Forest — all achieving the highest test accuracy of 77.71%. The K-Nearest Neighbors (KNN) model closely followed at 77.56%, while the Support Vector Machine (SVM) trailed slightly at 77.06%.

These seemingly strong accuracy values, however, are heavily influenced by the imbalanced nature of the response variable. In the test dataset, ‘NO INDICATION OF INJURY’ comprises 5,880 out of 7,944 observations (approximately 74% similar as the training dataset), creating a high baseline for models that primarily learn to predict this dominant class. In contrast, the rarest and most critical categories, ‘FATAL’

and ‘INCAPACITATING INJURY,’ appear only 17 and 332 times, respectively — representing a combined total of just 4.4% of the test set. This imbalance means that a model can achieve high overall accuracy by predominantly predicting the majority class, while essentially ignoring the minority ones.

Thus, while overall accuracy provides a useful high-level performance benchmark, it does not reflect the model’s effectiveness in identifying severe or rare injury outcomes and significant causes. These are often of the greatest interest in traffic safety analysis. To that end, accuracy must be interpreted alongside class-level metrics such as sensitivity, specificity, and F1 score, which are essential to understanding how well each model handles minority classes and supports injury prevention efforts in real-world applications.

Table 5-1 – General Accuracy for Each Model

Model	Accuracy
KNN	0.7756
LASSO	0.7771
GAM	0.7771
Decision Tree	0.7771
Random Forest	0.7771
SVM	0.7706

5.1.2 Each Class Performance

To better understand each model’s behavior beyond overall accuracy, Table 5-2 to Table 5-5 presents class-level evaluation metrics — sensitivity, specificity, and F1 score — for each of the four injury severity categories in the test dataset. These metrics offer insight into how well each model identifies rare but important outcomes.

Fatal and Incapacitating Injuries:

Across all models, the classification of ‘FATAL’ and ‘INCAPACITATING INJURY’ events was entirely unsuccessful. Sensitivity for both classes is 0 across all models, and F1 scores are undefined or not available due to the models never predicting these classes. While specificity remains perfect (1.0), this only reflects that the models correctly labeled other cases as not being fatal or incapacitating. The complete absence of true positive predictions indicates a severe limitation of all models in detecting these minority classes, likely stemming from their extremely low representation in the training data (only 17 fatal and 332 incapacitating cases in the test set). This raises critical concerns for practical use, particularly in safety-focused applications where identifying such cases is vital.

No Indication of Injury (Majority Class):

All models performed well in detecting the dominant class ‘NO INDICATION OF INJURY.’ Sensitivity values were consistently high (around 98.3–98.8%), and F1 scores exceeded 0.87 for all models, demonstrating a strong balance between precision and recall. This result is expected, as this class

makes up nearly 74% of the test set, making it the easiest to learn. However, specificity — the ability to correctly identify non-cases — was poor (around 19–24%), revealing that the models often misclassified other injuries as 'NO INDICATION OF INJURY.'

Non-Incapacitating Injuries:

Performance was moderate for the 'NONINCAPACITATING INJURY' category. Sensitivity scores for most models hovered around 22.2%, and F1 scores were approximately 0.33. While still limited, these figures indicate that the models are at least partially able to detect this class, likely because it is the second most common category in the test set (1,715 observations). Interestingly, the SVM model underperformed on this class, with a lower sensitivity (18.1%) and F1 score (0.28), suggesting it was less effective in balancing the classification boundary for this mid-frequency class.

Table 5-2 –Performance Metrics of Models Predicting the Accident as Fatal Accident

Class: Fatal			
Model	**Sensitivity**	**Specificity**	**F1**
KNN	0	1	NA
LASSO	0	1	NA
GAM	0	1	NA
Decision Tree	0	1	NA
Random Forest	0	1	NA
SVM	0	1	NA

Table 5-3 - Performance Metrics of Models Predicting the Accident as Incapacitating Injury Accident

Class: INCAPACITATING INJURY			
Model	**Sensitivity**	**Specificity**	**F1**
KNN	0	0.99987	NA
LASSO	0	1	NA
GAM	0	1	NA
Decision Tree	0	1	NA
Random Forest	0	1	NA
SVM	0	1	NA

Table 5-4 - Performance Metrics of Models Predicting the Accident as No Injury Accident

Class: NO INDICATION OF INJURY			
Model	**Sensitivity**	**Specificity**	**F1**
KNN	0.982993	0.237888	0.873574
LASSO	0.984864	0.237403	0.874443
GAM	0.984864	0.237403	0.874443
Decision Tree	0.984864	0.237403	0.874443
Random Forest	0.984864	0.237403	0.874443
SVM	0.988435	0.193314	0.870255

Table 5-5 - Performance Metrics of Models Predicting the Accident as Nonincapacitating Injury Accident

Class: NONINCAPACITATING INJURY

Model	Sensitivity	Specificity	F1
KNN	0.223324	0.966768	0.332321
LASSO	0.222741	0.968374	0.333043
GAM	0.222741	0.968374	0.333043
Decision Tree	0.222741	0.968374	0.333043
Random Forest	0.222741	0.968374	0.333043
SVM	0.180758	0.974795	0.284143

5.1.3 R Code

```r
# --- 4.0 --- Test
get_class_metrics <- function(model_name, pred, truth) {
  cm <- confusionMatrix(pred, truth)
  acc <- as.numeric(cm$overall["Accuracy"])
  by_class <- as.data.frame(cm$byClass)

  # For binary classification, convert vector to one-row data frame
  if (is.null(dim(by_class))) {
    by_class <- as.data.frame(t(by_class))
    rownames(by_class) <- levels(truth)[2]
  }

  by_class$Model <- model_name
  by_class$Class <- rownames(by_class)
  by_class$Accuracy <- acc

  by_class %>%
    select(Model, Class, Accuracy, Sensitivity, Specificity, F1)
}

# --- Generate predictions and metrics for each model ---
accuracy_list <- list()
# KNN
pred_knn <- predict(knn_model, newdata = df_test)
knn_metrics <- get_class_metrics("KNN", pred_knn, df_test$most_severe_injury)
cm_knn <- confusionMatrix(pred_knn, df_test$most_severe_injury)
accuracy_list[["KNN"]] <- cm_knn$overall["Accuracy"]

# LASSO
x_test <- model.matrix(~ ., data = df_test[, -ncol(df_test)])[ , -1]
pred_lasso_prob <- predict(lasso_model, newx = x_test, s = lasso_model$lambda.min, type = "response")
pred_lasso <- colnames(pred_lasso_prob)[apply(pred_lasso_prob, 1, which.max)]
pred_lasso <- factor(pred_lasso, levels = levels(df_test$most_severe_injury))
lasso_metrics <- get_class_metrics("LASSO", pred_lasso, df_test$most_severe_injury)
cm_lasso <- confusionMatrix(pred_lasso, df_test$most_severe_injury)
accuracy_list[["LASSO"]] <- cm_lasso$overall["Accuracy"]

# GAM
pred_gam_prob <- predict(final_gam, newdata = df_test, type = "response")
pred_gam <- colnames(pred_gam_prob)[apply(pred_gam_prob, 1, which.max)]
pred_gam <- factor(pred_gam, levels = levels(df_test$most_severe_injury))
gam_metrics <- get_class_metrics("GAM", pred_gam, df_test$most_severe_injury)
cm_gam <- confusionMatrix(pred_gam, df_test$most_severe_injury)
accuracy_list[["GAM"]] <- cm_gam$overall["Accuracy"]


# Decision Tree
pred_tree <- predict(tree_pruned, newdata = df_test, type = "class")
tree_metrics <- get_class_metrics("Decision Tree", pred_tree, df_test$most_severe_injury)
cm_tree <- confusionMatrix(pred_tree, df_test$most_severe_injury)
accuracy_list[["Decision Tree"]] <- cm_tree$overall["Accuracy"]
```

```r
# Random Forest
pred_rf <- predict(rf_model, newdata = df_test)
rf_metrics <- get_class_metrics("Random Forest", pred_rf, df_test$most_severe_injury)
cm_rf <- confusionMatrix(pred_rf, df_test$most_severe_injury)
accuracy_list[["Random Forest"]] <- cm_rf$overall["Accuracy"]

# SVM
pred_svm <- predict(svm_model, newdata = df_test)
svm_metrics <- get_class_metrics("SVM", pred_svm, df_test$most_severe_injury)
cm_svm <- confusionMatrix(pred_svm, df_test$most_severe_injury)
accuracy_list[["SVM"]] <- cm_svm$overall["Accuracy"]

# --- Combine all metrics into a single data frame ---
class_results <- bind_rows(
  knn_metrics, lasso_metrics, gam_metrics,
  tree_metrics, rf_metrics, svm_metrics
)

class_results <- round(class_results, 4)

# -- General Performance Metrics ---
accuracy_df <- data.frame(
  Model = names(accuracy_list),
  Accuracy = round(as.numeric(unlist(accuracy_list)), 4)
)


# --- Split and display by class ---
split_by_class <- split(class_results, class_results$Class)

for (class_name in names(split_by_class)) {
  cat("\n=== Metrics for Class:", class_name, "===\n")
  print(split_by_class[[class_name]])
}
```

5.2 Model Selection and Fit to all

5.2.1 Model Selection

Based on the comparative evaluation of multiple classification models, including both overall accuracy and class-level performance metrics, the LASSO Multinomial Logistic Regression model is selected as the most appropriate model for this analysis.

From the general performance results, LASSO achieved the highest accuracy (77.71%), slightly outperforming or matching other models such as GAM, Decision Tree, and Random Forest. However, the marginal differences in accuracy are not sufficient alone to determine model superiority, especially given the class imbalance present in the dataset.

What sets LASSO apart is its interpretability and its comparable or identical performance to more complex models. In the class-specific evaluation, LASSO matched other top-performing models in sensitivity and F1 score for the dominant class ‘NO INDICATION OF INJURY’ and demonstrated similar, albeit limited, ability to detect the ‘NONINCAPACITATING INJURY’ class. Like all models, it failed to predict ‘FATAL’ and ‘INCAPACITATING INJURY’ cases, reflecting the need for future strategies to address class imbalance.

Importantly, the LASSO model offers clear variable coefficients, as introduced in section 4.2.3 of the LASSO training results, enabling direct inference about the direction and magnitude of predictors. This makes it especially valuable in public safety applications where transparency and explanation of risk

factors are critical. Moreover, LASSO inherently performs variable selection through regularization, helping to reduce overfitting and improving generalization.

5.2.2 Fit to the Whole Dataset

The finalized LASSO multinomial logistic regression trained before was fit to the complete cleaned dataset that with 39726 observations. The selected regularization strength ($\lambda = 0.0305$) was chosen based on 10-fold cross-validation to minimize classification error while balancing model complexity.

```
# --- Fit to all data ---
x_all <- model.matrix(~ ., data = df_clean[, -ncol(df_clean)])[ , -1]


pred_probs <- predict(lasso_model, newx = x_all, s = lasso_model$lambda.min, type = "response")
pred_classes <- colnames(pred_probs)[apply(pred_probs, 1, which.max)]
pred_classes <- factor(pred_classes, levels = levels(df_clean$most_severe_injury))

conf_mat <- confusionMatrix(pred_classes, df_clean$most_severe_injury)
```

5.2.3 Interpretation and Weakness

The overall accuracy of the model on the full dataset was 77.5%, which exceeds the No Information Rate (NIR) of 74.0%. The NIR or null accuracy represents the accuracy achieved by always predicting the majority class ('NO INDICATION OF INJURY'). A statistically significant p-value (< 2.2e-16) indicates that the LASSO model provides meaningful predictive power beyond this naive baseline. The Kappa statistic, which accounts for class imbalance and agreement beyond chance, was moderate at 0.24, further validating the model's effectiveness in distinguishing among injury levels.

Despite an acceptable overall performance, the confusion matrix, shown in Table 5-6,reveals several important class-specific limitations. The model performed exceptionally well on the majority class, 'NO INDICATION OF INJURY', achieving a sensitivity of 98.5% and precision (positive prediction value) of 78.4%, with an F1 score near 0.87. However, performance deteriorated significantly for other categories. The model captured only 21.3% of 'NONINCAPACITATING INJURY' cases (sensitivity), though it maintained a high specificity of 96.9% for that class.

Critically, the model failed to predict any instances of the 'FATAL' and 'INCAPACITATING INJURY' categories. Both categories had a sensitivity of 0.0%, despite perfect specificity (i.e., the model never misclassified other classes as these). This outcome stems from the extreme class imbalance observed in the training dataset: fatal injuries represented only 0.2% of the training data (87 cases out of over 39,000, shown in Table 4-2), while incapacitating injuries accounted for approximately 4.2%. As a result, the model effectively learned to ignore these rare but critically important outcomes during prediction.

In conclusion, the LASSO model is highly effective at predicting non-severe injury outcomes, particularly the majority class. However, its failure to identify severe injuries limits its applicability in safety-critical contexts without further adjustments to handle class imbalance.

While LASSO's interpretability and variable selection capabilities are strengths, future work should consider techniques such as class weighting, resampling, or alternative models more robust to imbalanced classes to improve performance for rare but important outcomes like fatal injuries.

Notably, the fitted LASSO model selected a minimal set of predictors, with non-zero coefficients concentrated exclusively under the 'NO INDICATION OF INJURY' class shown in Table 4-3. This pattern indicated that the severe class imbalance has a strong influence on feature selection. Predictors such as crash types—especially 'REAR END', 'SIDESWIPE SAME DIRECTION', 'PEDESTRIAN', and 'PEDALCYCLIST'—played a key role in differentiating low-risk outcomes. This reinforces the conclusion that while the model is reliable for predicting common outcomes, it currently lacks sufficient discriminatory power for rare but critical injury types due to limited signal from those classes. Thus, the final model reflects strong predictive utility for majority outcomes but highlights the need for future work to enhance representation and learning for severe cases.

Table 5-6 – Confusion Matrix Result for LASSO Model General Performance on the Whole Dataset

		Reference			
		FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
Prediction	FATAL	0	0	0	0
	INCAPACITATING INJURY	0	0	0	0
	NO INDICATION OF INJURY	65	1163	28957	6745
	NONINCAPACITATING INJURY	22	500	444	1830

Table 5-7 – LASSO Model Overall Statistics

Accuracy	0.775
95% CI	(0.7708, 0.7791)
No Information Rate	0.7401
P-Value [Acc > NIR]	2.20E-16
Kappa	0.2419

Table 5-8 – LASSO Model Statistics by Class

	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
Sensitivity	0	0	0.9849	0.21341
Specificity	1	1	0.2278	0.96899
Pos Pred Value	NaN	NaN	0.7841	0.65451
Neg Pred Value	0.9978	0.95814	0.8412	0.81736
Prevalence	0.0022	0.04186	0.7401	0.21585
Detection Rate	0	0	0.7289	0.04607
Detection Prevalence	0	0	0.9296	0.07038
Balanced Accuracy	0.5	0.5	0.6063	0.5912
F1 Score	NaN	NaN	0.8731	0.3219

To better understand the modeling limitations observed in previous sections—particularly the models' consistent failure to correctly predict FATAL and INCAPACITATING INJURY cases—it is helpful to examine how the categorical predictors are distributed across injury severity levels. Using cross-tabulations and standardized residuals from chi-squared tests, we evaluated the relationships between injury severity and variables such as crash hour, weather condition, lighting condition, first crash type, and others.

Across nearly all categorical variables, the FATAL category consistently has very low counts. For example, in the lighting condition variable, only 1 crash under 'DARKNESS,' 2 under 'DAWN,' and 4 under 'DUSK' resulted in fatal injuries, compared to 37 under 'DAYLIGHT' and 43 under 'DARKNESS, LIGHTED ROAD.' While the absolute numbers are already low, the standardized residuals from the chi-squared test further reveal that these values rarely reach significance thresholds (e.g., $|\text{residual}| > 2$), meaning fatal crashes are not strongly over- or under-represented in any specific category. This lack of contrast reduces the model's ability to learn distinct patterns that are predictive of fatal outcomes.

A similar pattern is observed for incapacitating injuries. While this class has higher representation than fatal injuries, its counts remain modest (e.g., 46 under 'DUSK,' 62 under 'DARKNESS,' 47 under 'DAWN') and spread thinly across levels. The standardized residuals do show some strong associations—such as overrepresentation of incapacitating injuries under 'DARKNESS, LIGHTED ROAD' and 'DAWN'—but these patterns are not consistent or strong across all variables. Additionally, variables like crash_day_of_week, traffic_control_device, and weather_condition exhibit relatively small or statistically insignificant deviations in fatal and incapacitating cases, making it harder for models to detect meaningful class distinctions.

This weak association is compounded by the overwhelming dominance of the 'NO INDICATION OF INJURY' category in all variables. For instance, in nearly every level of lighting, weather, or crash type, this class accounts for most observations, frequently producing strong positive residuals (e.g., +6.03 for 'DAYLIGHT' and +23.15 for 'REAR END'), indicating that the data is biased toward safe or non-injurious outcomes. As a result, predictive models are inclined to learn rules that favor this majority class, and tend to ignore minority outcomes.

In sum, the modeling challenges stem not only from response imbalance, but also from predictor distributions that do not sufficiently differentiate between rare injury categories. Variables either lack enough fatal/incapacitating observations altogether or show too little variation to yield strong predictive patterns. This reinforces the importance of collecting more balanced data or using strategies such as class weighting, synthetic oversampling (e.g., SMOTE), or cost-sensitive modeling to improve classification performance for underrepresented but critical outcomes.

Table 5-9 – Crash Hour Sample Distribution

Crash Hour	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
AM Peak	6	203	4449	1219
PM Peak	11	308	6047	1677
Off-Peak	39	799	14166	3978
Night	31	353	4739	1701

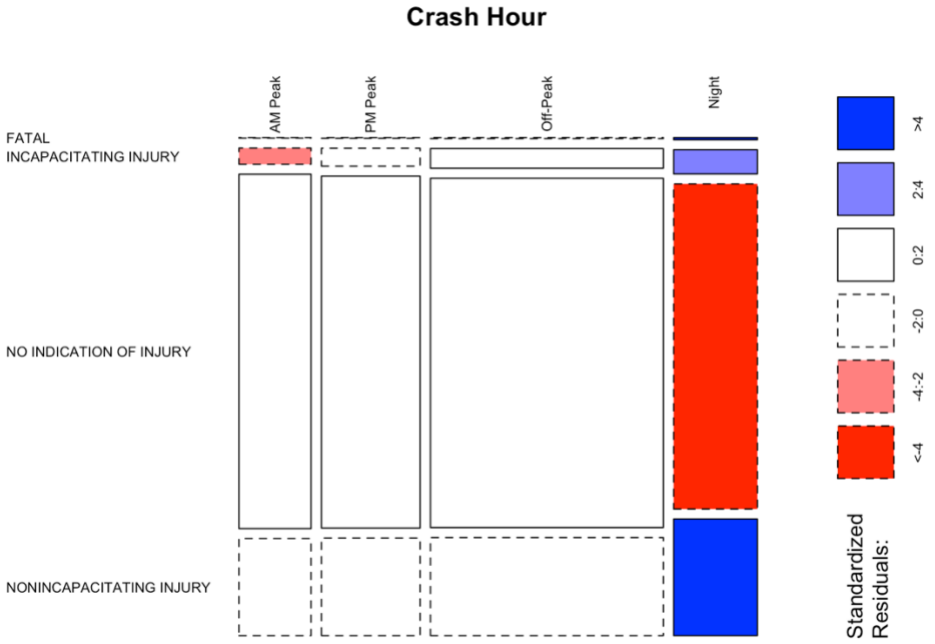

Figure 5-1Crash Hour Chi-squared Test Plot

Table 5-10 – Crash Month Sample Distribution

crash_month	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
1	5	99	2144	539
2	5	95	1899	441
3	4	98	2020	586
4	4	126	2142	584
5	11	137	2635	804
6	6	188	2665	871
7	15	182	2644	851
8	7	144	2604	866
9	5	161	2698	825
10	8	179	2767	819
11	12	134	2549	682
12	5	120	2634	707

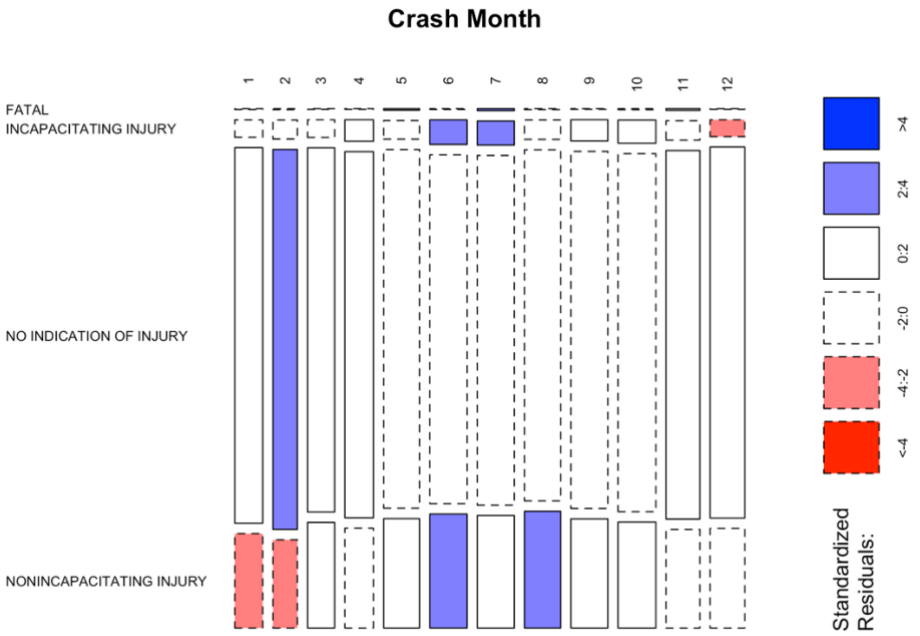

Figure 5-2 - Crash Month Chi-squared Test Plot

Table 5-11 – Crash Day of Week Sample Distribution

Crash_day_of_ Week	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
Weekday	65	1135	20259	5987
Weekend	22	528	9142	2588

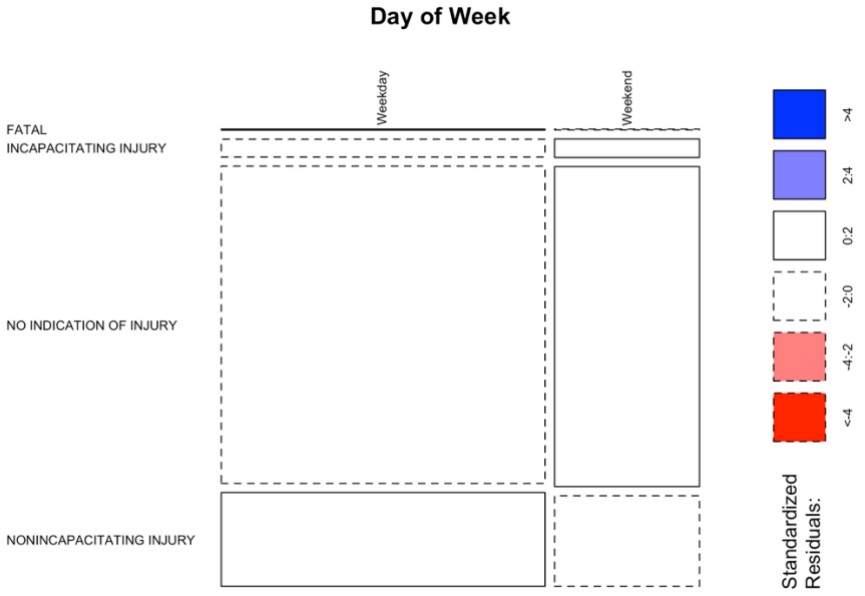

Figure 5-3Day of Week Chi-squared Test Plot

Table 5-12- Traffic Control Device Sample Distribution

traffic_control_device	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
NO CONTROLS	4	159	2772	681
STOP SIGN/FLASHER	19	390	8671	2536
TRAFFIC SIGNAL	64	1114	17958	5358

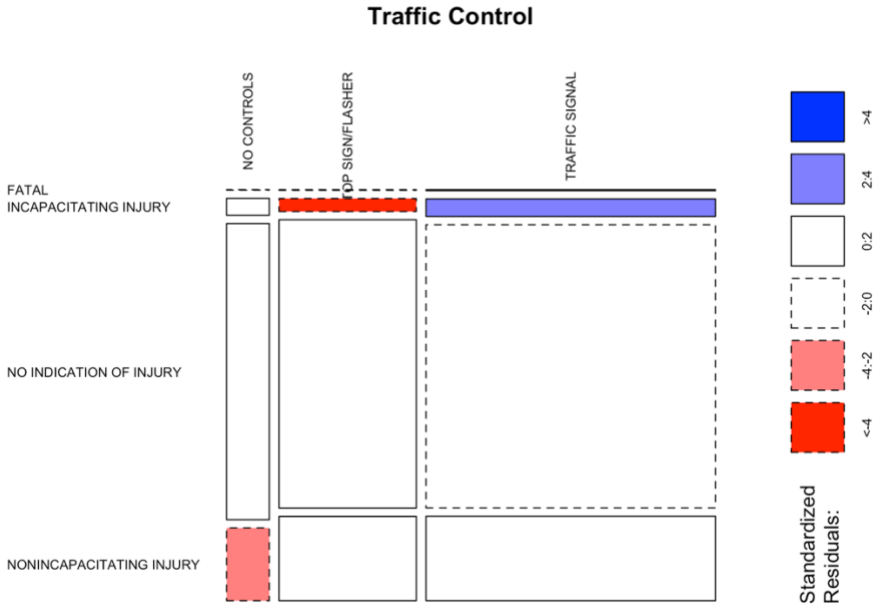

Figure 5-4 Traffic Control Chi-squared Test Plot

Table 5-13 – Weather Condition Sample Distribution

weather_condition	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
CLEAR	71	1447	24377	7199
CLOUDY/OVERCAST	6	45	1156	298
RAIN	8	148	3038	879
SNOW	2	23	830	199

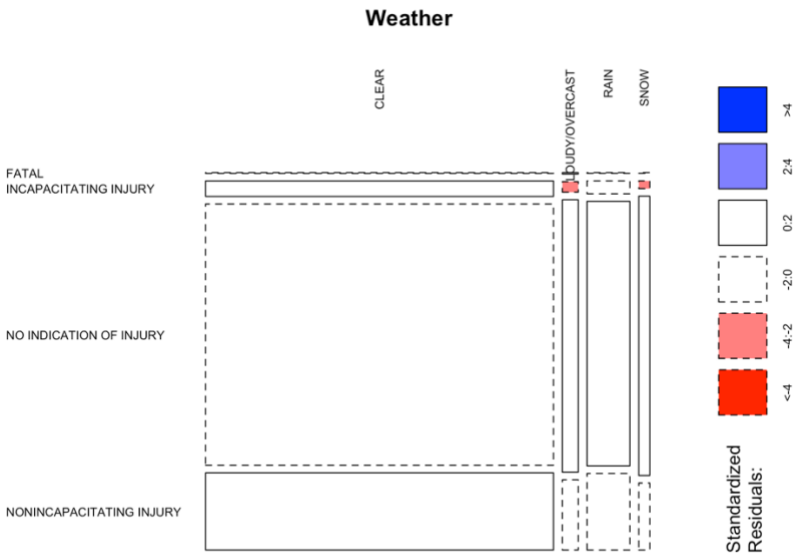

Figure 5-5 Weather Condition Chi-squared Test Plot

Table 5-14 – Lighting Condition Sample Distribution

lighting_condition	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
DARKNESS	1	62	833	235
DARKNESS, LIGHTED ROAD	43	524	7824	2512
DAWN	2	47	566	169
DAYLIGHT	37	984	19298	5416
DUSK	4	46	880	243

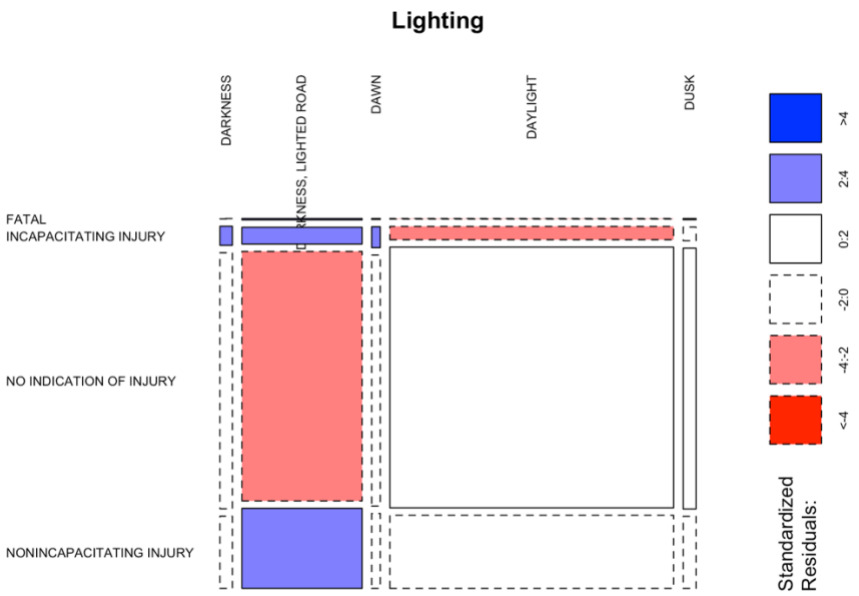

Figure 5-6 Lighting Condition Chi-squared Test Plot

Table 5-15 – First Crash Type Sample Distribution

first_crash_type	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
ANGLE	36	634	9893	3589
FIXED OBJECT	2	37	665	170
PARKED MOTOR VEHICLE	3	10	524	44
PEDALCYCLIST	4	153	273	707
PEDESTRIAN	18	347	171	1123
REAR END	0	78	4394	556
SIDESWIPE SAME DIRECTION	2	16	2204	137
TURNING	22	388	11277	2249

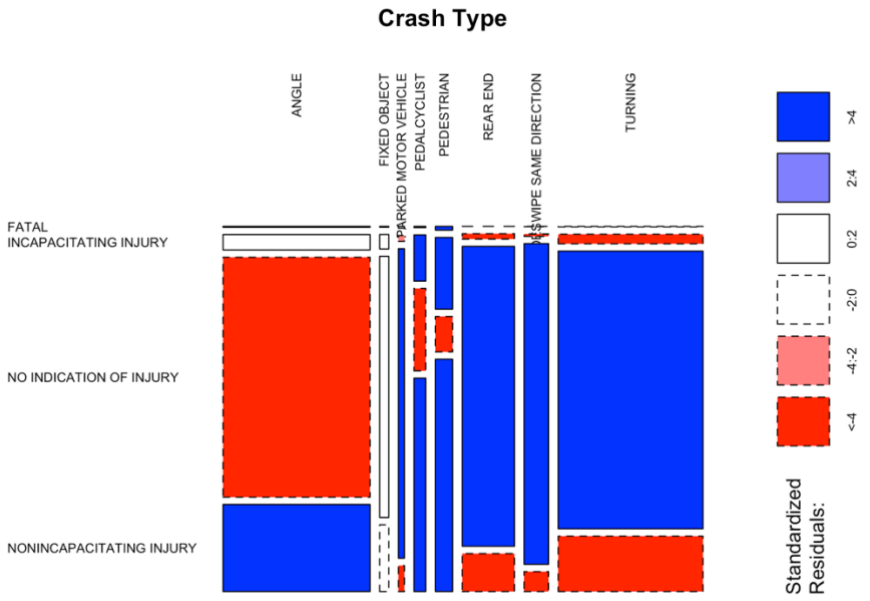

Figure 5-7 Crash Type Chi-squared Test Plot

Table 5-16 – Roadway Surface Condition Sample Distribution

roadway_surface_condition	FATAL	INCAPACITATING INJURY	NO INDICATION OF INJURY	NONINCAPACITATING INJURY
DRY	73	1432	24140	7093
SNOW OR SLUSH	2	17	753	152
WET	12	214	4508	1330

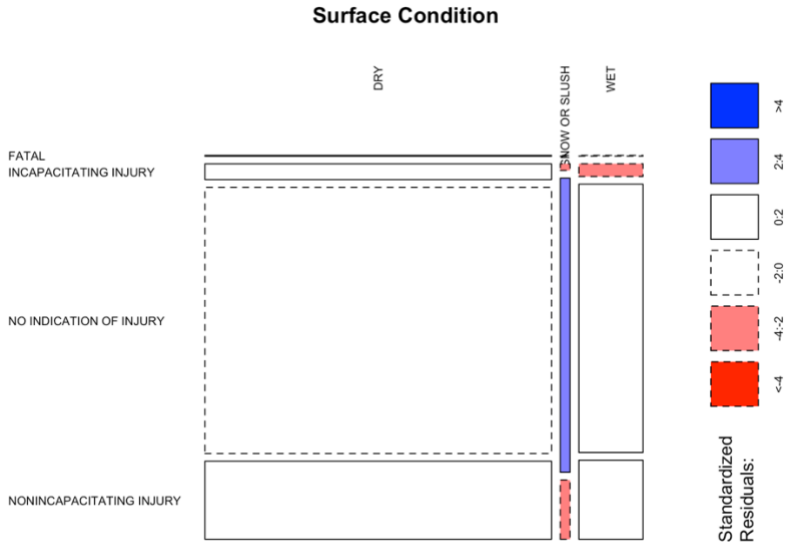

Figure 5-8 Surface Condition Chi-squared Test Plot

5.2.4 R Code

```r
# --- vairables distribution checker ---
chisq.test(table(df_clean$crash_hour, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$crash_hour, df_clean$most_severe_injury),
           main = "Crash Hour", shade = TRUE, las = 2)

chisq.test(table(df_clean$crash_day_of_week, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$crash_day_of_week, df_clean$most_severe_injury),
           main = "Day of Week", shade = TRUE, las = 2)

chisq.test(table(df_clean$traffic_control_device, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$traffic_control_device, df_clean$most_severe_injury),
           main = "Traffic Control", shade = TRUE, las = 2)

chisq.test(table(df_clean$weather_condition, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$weather_condition, df_clean$most_severe_injury),
           main = "Weather", shade = TRUE, las = 2)

chisq.test(table(df_clean$lighting_condition, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$lighting_condition, df_clean$most_severe_injury),
           main = "Lighting", shade = TRUE, las = 2)

chisq.test(table(df_clean$first_crash_type, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$first_crash_type, df_clean$most_severe_injury),
           main = "Crash Type", shade = TRUE, las = 2)

chisq.test(table(df_clean$roadway_surface_cond, df_clean$most_severe_injury))$stdres
mosaicplot(table(df_clean$roadway_surface_cond, df_clean$most_severe_injury),
           main = "Surface Condition", shade = TRUE, las = 2)

chisq.test(table(as.factor(df_clean$crash_month), df_clean$most_severe_injury))$stdres
mosaicplot(table(as.factor(df_clean$crash_month), df_clean$most_severe_injury),
           main = "Crash Month ", shade = TRUE, las = 2)
```

5.3 Lessons Learned

This project has been both a challenging and rewarding experience, offering me valuable insights into data-driven modeling with categorical outcomes, and strengthening my understanding of both statistical methods and practical data limitations.

5.3.1 Deepened Understanding of Qualitative Modeling

Throughout the coursework and textbook exercises, most examples focused heavily on quantitative regression modeling. In contrast, this project centered on predicting a qualitative, multiclass outcome (injury severity) using mostly categorical predictors. Navigating through this unfamiliar territory with methods such as multinomial logistic regression, classification trees, and support vector machines forced me to confront the unique challenges of classification modeling. While I encountered considerable difficulty—particularly in tuning models and interpreting results—the process significantly enhanced my comprehension of how qualitative models are structured and trained.

5.3.2 Reinforced the Value of Understanding the Data First

One of the most instructive insights from this project emerged during the initial modeling phase. Despite careful model development and parameter tuning, most models failed to accurately predict the rare but critical injury outcomes, such as FATAL and INCAPACITATING INJURY. Furthermore, the initial decision tree model, which selected only a single predictor, emphasized the limitations imposed by the data itself. These outcomes prompted a deeper investigation into the data structure, particularly the distribution of both the response and predictor variables. Chi-squared tests and standardized residual

analysis revealed strong imbalances in the response variable and a lack of distinguishing patterns in many of the predictors. These findings underscore the importance of thoroughly understanding the data before proceeding to model selection or tuning. A robust understanding of data distribution is not only essential for interpreting model outcomes but also for guiding model construction strategies. For instance, recognizing the imbalance in the response variable early on could have informed the use of resampling techniques—such as oversampling minority classes or applying class weights—to improve the model's sensitivity to rare outcomes. The absence of such adjustments was a key limitation of this project, highlighting how data diagnostics should directly inform modeling decisions to enhance both performance and fairness.

5.3.3 Broadened Perspective in Transportation Safety Applications

As a transportation engineering student, this project has also inspired me to think more critically about how injury severity is categorized in traffic safety studies. Transportation agencies, such as NCDOT, use a more nuanced classification system for crash severity (e.g., K - fatal, A - severe, B - moderate, C - possible injury, and PDO - property damage only). A richer classification scheme can better capture the diversity of crash outcomes and may help reduce imbalance in the response variable, ultimately supporting the construction of more robust and interpretable models. This reinforces the idea that thoughtful variable design—in both response and predictor dimensions—plays a crucial role in successful modeling, particularly in safety-critical domains.