CIS 5200 Machine Learning - Final Project Report

Predicting Traffic Accident Risk from Local Music Listening Patterns

Mateo Taylor, Lucas Flahault, Steyn Knollema

December 7, 2025

1 Abstract

Traffic accidents remain a leading cause of injury and death in the United States, influenced by a complex web of environmental, temporal, and human factors. While distracted driving is a known risk factor, the specific influence of ambient auditory environments, specifically popular music trends, on accident severity remains under-researched. This study investigates the predictive power of Spotify audio features (e.g., tempo, energy, danceability) on traffic accident severity. We integrated a dataset of 4.1 million geolocated U.S. traffic accidents with weekly regional Spotify Top 200 charts rankings. We evaluated a suite of machine learning models, ranging from Linear Regression & Kernel methods to Ensemble and Neural Network methods, comparing performance on a baseline dataset against an augmented dataset containing musical features. Our results indicate that tree-based ensemble methods, particularly XGBoost, significantly outperform linear models and neural networks on this tabular data. While the inclusion of musical features yielded a statistically positive increase in predictive accuracy (boosting XGBoost R^2 from 0.4152 to 0.4352 post-tuning), the marginal gain suggests that music listening patterns serve as a proxy for latent behavioral variables rather than a primary determinant of severity.

Final Project IPYNB:
https://drive.google.com/file/d/1gnfsMHXWM0NOQck57K6xmyjtnC8Dn$_sb/view?usp = sharing$

2 Motivation

In a country as large and geographically diverse as the United States, traffic accident prevention methods benefit from understanding what conditions can increase the severity of traffic accidents. While research has extensively studied phone use and intoxication, the relationship between in-vehicle music and accident risk remains largely unexplored, despite 68% of drivers regularly listening to music while driving [1]. Traffic often prevents first responders from navigating to an accident scene, leaving ambulances stranded and unable to treat patients. Our goal was to combine geocoded US traffic accident data with Spotify listening data between 2017-2021, with the goal of predicting how severely an accident occurrence will impact traffic flow.

3 Related Work

A majority of existing statistics and machine learning work focuses on forecasting recurrent traffic patterns, rather than traffic incidents caused by non-recurrent events like accidents. Traffic forecasting often treats accidents as an outcome of poor traffic conditions, rather than using them as

predictors of traffic [2]. The authors of the dataset used in this project, Moosavi et al. [4], use their data in combination with another dataset of traffic incidents to predict if accidents will occur during a given traffic event in a small selection of cities. Another related project we found was done by Talluri et. al. [6], who uses Bayesian Networks to create explainable clusters of road features that predict how an accident might impact traffic severity. Their research also augments a version of Moosavi's dataset with geographic simulations of the accident environment. Importantly, they highlight how relationships in traffic data are often non-linear, and that simple non-linear models can often achieve strong performance while remaining interpretable.

4 Dataset

4.1 Primary Dataset: US Traffic Accidents (2016-2023) [5]

- **Source:** Kaggle - `https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents`
- **Size:** $n = 7.73$ million records, $p = 47$ features, (approx. 4.08 million after processing and merging with Spotify data)
- **Coverage:** 49 US states, 2016-2023
- **Key Features:** Severity (1-4), timestamp, location (city, state, lat/long), weather conditions (temperature, precipitation, visibility), road features (junction, traffic signal, crossing), environmental factors

4.2 Spotify Dataset: Spotify Charts Weekly Top 200 (2017-2021)

- **Source:** Github - `https://github.com/younver/spotify-top-200-dataset`
- **Size:** $n = 74,661$ songs, $p = 40$ features
- **Coverage:** US, 2017-2021
- **Key Features:** Spotify's song attributes: (Danceability, Energy, Key, Mode, Time Signature, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Duration)

4.3 Feature Processing

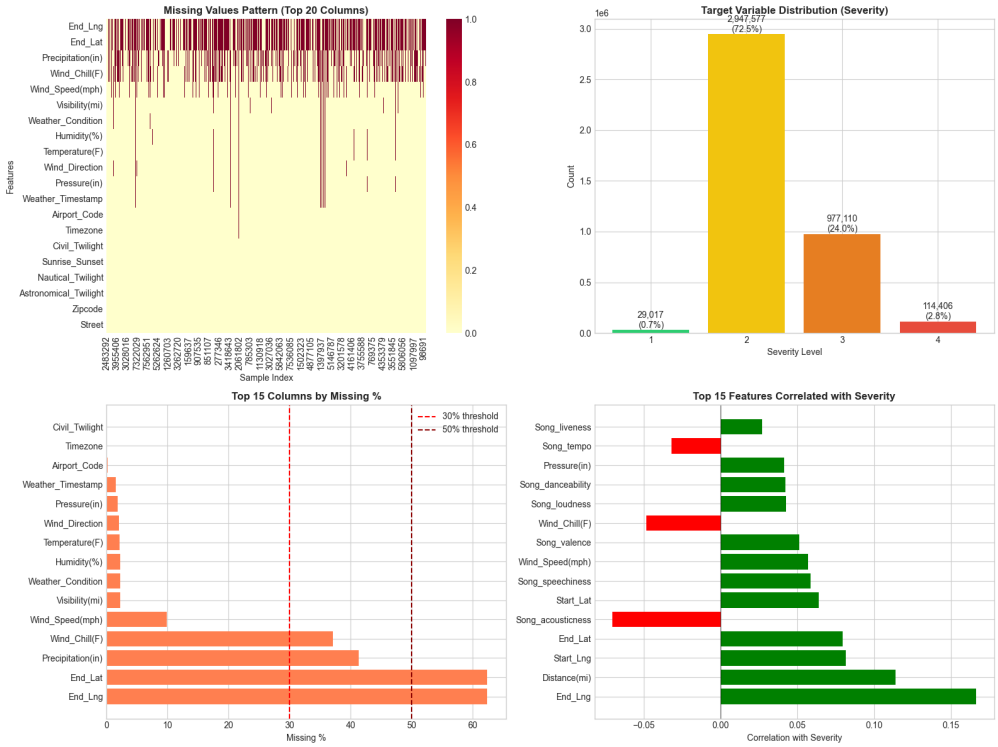

Figure 1: Data Quality visualisation showing the distribution of the target variable (Severity) across the traffic dataset, Missing Values patterns, Top missing Colums, and top 15 features correlated with Severity. The class imbalance between minor (1-2) and severe (3-4) accidents informed our choice of evaluation metrics.

Additional Features: We expanded StartTime and EndTime into a set of features that better indicate the time of day that the accident occurred, including *Accident_Duration*, *Start_Hour*, *Day*, *Month*, *Is_Morning_Rush_Hour*, *Is_Evening_Rush_Hour*, and *Is_Night*. We then added in the Spotify data by mapping each accident to the top song of the week.

Removed Features: While our dataset was relatively clean, we chose to exclude 4 features with more than 30% of their entries missing: *End_Lat*, *End_Lng*, *Wind_Chill(F)*, and *Precipitation(in)*. Additionally, we removed a number of features from the accidents dataset that we felt had too much variance to be useful predictors or were redundantly encoded in other variables: *ID, Source, Description, Street, County, Zipcode, Country, Airport_Code, Weather_Timestamp.*

5 Problem Formulation

Our project is a supervised regression, where we attempt to predict a traffic severity score (between 1 and 4) for a given accident. This severity score is a measure of how much an accident impacts traffic flow, with a 1 being low or no impact and 4 being a severe impact on traffic. We formulated the problem as a supervised regression task rather than classification because accident severity is an ordinal hierarchy (from 1 to 4) rather than simple independent categories. Treating the targets as a continuous score ensures that the loss function puts higher penalties on large deviations. Through

this, we aimed to preserve the magnitude of the traffic impact. Our dataset was large with relatively few features, which means that the risk of overfitting was relatively low. Linear regression seemed like a strong starting point, as we didn't expect regularization techniques like ridge regression and lasso to significantly improve model performance. After our baseline linear model, we tried to incorporate Kernel methods to transform our (non linear) dataset into higher dimensions so that it might be linearly separable, then compared performance with tree-based models (XGBoost, Adaboost, Random Forest) and neural networks.

We prioritized building an effective model without our Spotify features, which we then compared to a full model. While we are interested in seeing how music patterns impact accident severity, isolating a baseline model helps to avoid overstating their effect. This approach allows us to more easily interpret how much music features actually improve the model.

6 Methods

6.1 Initial Thoughts & Hypotheses

Given the structured and tabular nature of the accident and Spotify datasets, we formulated specific hypotheses regarding model performance prior to experimentation.

We hypothesized that tree-based ensemble methods (specifically Random Forest, Adaboost, and Gradient Tree Boosting) would yield the highest predictive accuracy. Tabular data often contains irregular decision boundaries and mixed feature types (categorical and numerical) that decision trees naturally handle well. We expected these models to effectively capture non-linear interactions between environmental variables (e.g. weather, time, temperature) and musical features without requiring extensive feature scaling.

Conversely, we anticipated that Neural Networks (NNs) would underperform in this specific context. While NNs excel at extracting features from unstructured, high-complexity data, our dataset consists of distinct, semantic features with relatively low hierarchical complexity. We predicted that deep learning architectures would likely struggle to find optimal convergence or would overfit the noise in the data, failing to outperform the efficient, greedy splitting algorithms found in gradient boosting machines. However, we still wanted to test our hypothesis and ran NN models to evaluate our beliefs.

Initial trial through several different models Due to our dataset being (relatively) simple (and Steyn wanting to stress test his new laptop) we tried running many different models on a 500,000 datapoint subset of the full dataset. During this, we used a small GridSearchCV to find the optimal hyperparameter values without optimizing the tested methods too much yet. This resulted in the results seen in the table below. This input was then used to decide on which models to run the full datapoints dataset on. Using this method we reduced compute time while increasing iterations and trials. The following models were tested and the accuracy results are shown next to it.

Model	Test MSE	Test RMSE	Test MAE	Test R^2	Train Time (s)
XGBoost	0.155941	0.394894	0.273067	0.435212	39.942470
LightGBM (HistGradient)	0.184345	0.429354	0.311239	0.332340	33.028278
Random Forest	0.185391	0.430571	0.303564	0.328550	306.603313
Decision Tree	0.193852	0.440286	0.306431	0.297909	72.192503
NN Simple (Adaptive Small)	0.195562	0.442224	0.314205	0.291715	758.600000
Linear Regression	0.245770	0.495752	0.394453	0.109870	8.545410
Ridge Regression	0.245782	0.495764	0.394454	0.109829	7.173924
Nyström (Kernel) + Ridge	0.248230	0.498227	0.395115	0.100963	14.949245
AdaBoost (DT)	0.252586	0.502579	0.433407	0.085185	613.900000
Lasso Regression	0.253060	0.503050	0.407467	0.083470	4.614527
NN Medium (Adaptive)	0.295822	0.543895	0.409679	-0.071406	549.400000
NN Complex (Adaptive Large)	0.348070	0.589975	0.420429	-0.260639	994.000000

Table 1: Model performance comparison on the test set and cross validation (With Songs Dataset).

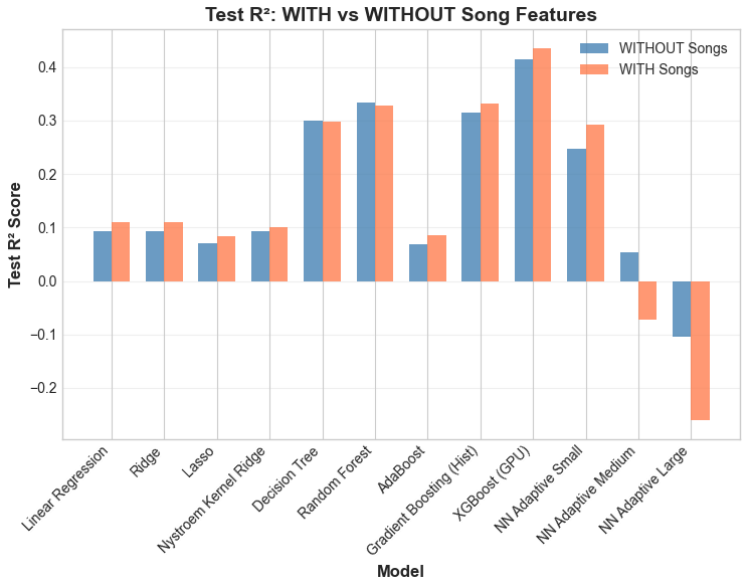

Figure 2: The performance metrics of running several (lightweight) models on all datapoints.

6.2 Baseline (Linear) models

We established Linear Regression as our fundamental baseline to benchmark the performance of more complex algorithms. To rigorously test the limits of linear relationships in our data, we extended this baseline to include regularized variants.

Ordinary Least Squares (OLS)

We implemented standard OLS regression using Scikit-learn to model the target as a linear combination of features. This served as a sanity check for feature directionality and a reference point for quantifying the value of non-linear modeling.

Ridge (L2) and Lasso (L1) Regression

- **Ridge Regression:** Applies an L2 penalty to the loss function, shrinking coefficients toward zero which would be particularly effective at handling multicollinearity among correlated environmental features.
- **Lasso Regression:** Applies an L1 penalty, which can shrink coefficients to exactly zero. We utilized Lasso to test if an explicit feature selection mechanism would improve the signal-to-noise ratio in the linear baseline.

For both Ridge and Lasso, we performed hyperparameter tuning via `GridSearchCV` on the random training split (80% of the data), searching over the regularization strength α and selecting the value that minimized validation error. This procedure yielded a best $\alpha = 10.0$ for Ridge and $\alpha = 0.01$ for Lasso. We standardized all numerical features (e.g., *Temperature*, *Song_tempo*) prior to training, as regularization techniques are sensitive to the scale of input variables.

Kernel Ridge Regression for Non-Linear Approximation

While linear models provide a stable baseline, they are fundamentally limited by their inability to capture the non-linear interactions which are inherent in our traffic accident data (e.g., the compounding effect of rain and night). To address this without incurring the prohibitive computational cost of full kernel methods on millions of records, we employed Nyström kernels as our novel method which is an approximation technique to project input features into a higher-dimensional space.

- **Nyström Approximation:**Standard Kernel Ridge Regression scales cubically ($O(N^3)$), making it infeasible for our dataset size. We utilized the Nyström method to approximate the feature map of the Radial Basis Function (RBF) kernel. This allowed us to compute a low-rank approximation of the kernel matrix and apply efficient linear Ridge Regression in the transformed space.

For the Nyström + Ridge pipelines, we again used a simple `GridSearchCV` on the training data to select the Ridge regularization parameter α under the same random 80/20 train–test split. For the configuration, we used a 100-dimensional Nyström approximation (`n_components=100`) of the RBF kernel and selected a best $\alpha = 1.0$, ensuring that all linear and kernel baselines were evaluated under well-tuned configurations.
As detailed in Table 1, the Nyström + Ridge configuration ($R^2 \approx 0.100$) underperformed the standard Linear Regression baseline ($R^2 \approx 0.109$).

This result debunked our hypothesis that non-linear modeling was necessary, the loss, and the higher computational cost made this method undesirable to explore further.

6.3 Neural Nets

We implemented three fully connected Neural Network architectures of increasing capacity in TensorFlow. In contrast to fixed-size baselines, these models adapt their width to the number of input features d. The goal was to test whether higher capacity and adaptive width meaningfully improve performance on our relatively simple tabular datasets or mainly lead to overfitting.

NN Adaptive Small

This model is a shallow fully connected network whose hidden layer widths scale linearly with the input dimension d (roughly $2d \to d \to d/2$). It serves as a low-capacity neural baseline: simple

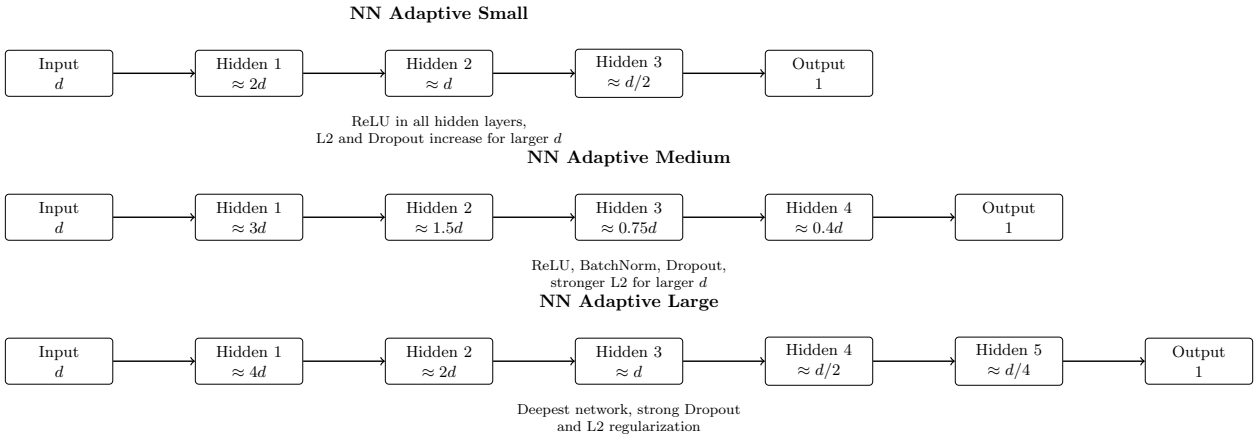

Figure 3: Adaptive feedforward neural networks. Width scales with the input dimension d. Small, Medium, and Large share the same basic pattern with ReLU hidden layers, BatchNorm and Dropout, and a single scalar output, with increasing depth and capacity.

enough to avoid severe overfitting on our relatively small tabular datasets, but flexible enough to capture basic nonlinear relationships beyond linear and tree-based models.

NN Adaptive Medium

NN Adaptive Medium increases both depth and width compared to the small model (approximately $3d \to 1.5d \to 0.75d \to 0.4d$), and applies Batch Normalization and stronger regularization. We use it to test whether a moderately higher-capacity network can exploit more complex interactions between features (in particular the added song features) and yield systematic performance gains over the small architecture.

NN Adaptive Large

NN Adaptive Large is our highest-capacity architecture, with five hidden layers (around $4d \to 2d \to d \to d/2 \to d/4$) and the strongest combination of Dropout and L2 regularization. It is designed as a "stress test" for model capacity: if additional complexity in the data is present and learnable, this model should be able to capture it; if not, its performance relative to the smaller networks reveals the extent to which large neural models simply overfit our setting.

All three architectures are trained with the Adam optimizer, early stopping, and learning-rate reduction on reaching a potential plateau. This shared training setup lets us attribute performance differences primarily to architectural capacity rather than optimization details.
Our resulting Neural Nets tended to overfit to noise in the training data, and NN Simple (as can be seen in the table above) performed the best out of all Neural Nets models. Neural Net complexity did not work well with our simple dataset, proving our hypothesis that it was not the best method to predict the severity of the accident and that it wouldn't be explored further.

6.4 Tree-based methods

Given the tabular nature of our data and the presence of non-linear relationships between environmental features and accident severity, we employed several tree-based algorithms.

Random Forest and Decision Trees

We utilized Random Forest regressors to reduce the variance associated with single Decision Trees. By averaging multiple deep decision trees, trained on different parts of the same training set, we aimed to create a robust baseline for non-linear modeling.

Gradient Boosting Machines

We explored several boosting algorithms, which build models sequentially to correct the errors of previous models. We tested:

- **AdaBoost:** Uses decision stumps as weak learners.
- **Histogram-based Gradient Boosting:** An optimized implementation of gradient boosting that bins input samples, significantly speeding up training on large datasets ($n > 10,000$).
- **XGBoost:** An optimized distributed gradient boosting library designed to be highly efficient and flexible. We utilized the squared error loss function and after a strong initial performance we used random search for further hyperparameter tuning. XGBoost was chosen as our primary candidate for optimization due to its superior handling of missing values and regularization capabilities (L1 and L2) which prevent overfitting.

6.5 Experiment Design

To rigorously test our hypothesis regarding music features, we designed a comparative experiment:

1. **Dataset A (Baseline):** Contains only temporal, environmental, and road-condition features ($p = 54$, derived from 46).
2. **Dataset B (Augmented):** Contains all features from Dataset A plus aggregated Spotify audio features such as *danceability*, *energy*, *Specific_Artist_Name*, *Specific_Song_Name* and *tempo* ($p = 82$, derived from 61).

We trained identical model architectures on both datasets of 4.1 million records and compared performance metrics (R^2, RMSE) to isolate the marginal contribution of the musical features.

7 Experiments and Results

With vs. Without Song Features

We executed a comparative analysis across all candidate models on the full processed dataset of 4.1 million records. The results demonstrated a consistent, albeit modest, improvement in model performance when song features were included.

As shown in Table 2, linear models saw the highest percentage improvement, likely because the song features provided variance not captured by the rigid linear assumptions of the baseline features. However, tree-based models offered significantly higher absolute performance. The tuned XGBoost model achieved an R^2 of 0.4352 with song features, surpassing the baseline of 0.4152.

Hyperparameter Tuning

Given that XGBoost was the top-performing algorithm, we performed a targeted Randomized Search Cross-Validation (25 iterations, 3-fold CV on training samples) on both Song and No_Song datasets, to optimize its hyperparameters on the baseline and augmented datasets. We tuned *n_estimators*, *max_depth*, *learning_rate*, and *reg_alpha/lambda*.

Model	R^2 (No Songs)	R^2 (With Songs)	ΔR^2	% Change
Linear Regression	0.0923	0.1099	+0.0175	+18.98%
Lasso	0.0700	0.0835	+0.0135	+19.32%
Decision Tree	0.3003	0.2979	-0.0024	-0.81%
Random Forest	0.3330	0.3286	-0.0044	-1.33%
(XGBoost (Both Tuned)	**0.4152**	**0.4352**	**+0.0200**	**+4.82%**

Table 2: Comparison of R^2 scores across datasets. While standard tree models (Random Forest, Decision Tree) saw a slight performance drop, boosting methods like XGBoost and linear models capitalized on the additional song feature information.

The tuning process yielded significant gains for both Song & No_Song databases. Both datasets resulted in roughly similar XGBoost parameters after doing an extensive RandomSearch on 1M datapoints for both datasets.

For the Songs database the following results were found

- **Optimal Parameters:** $n\_estimators = 3570$, $learning\_rate = 0.01959$, $max\_depth = 7$, $sample\_per\_tree = 0.7088$.
- **Final Test R^2: 0.4352**
- **Final Test RMSE: 0.3948**

This represents a substantial improvement over the baseline Random Forest ($R^2 \approx 0.29$) and baseline OLS ($R^2 \approx 0.11$) and a slight improvement over the No_Songs model ($R^2 = 0.4152$) .

Feature Importance Analysis

To understand why the models improved, we analyzed the feature importance of the final XGBoost model. While the location-based feature, *Traffic_signal*, remained the dominant predictors, musical features did appear in the top features.

- **Top Song Features:** *Song_track_name_DÁKITI (Song by Bad Bunny)*, *Song_artist_names_The Weekends*, *Song_speechiness*, and *Song_duration* were the most influential musical predictors.

This suggests that while music is not a primary driver of accidents, it acts as an environmental variable that can assist the model's ability to classify severity in marginal cases.

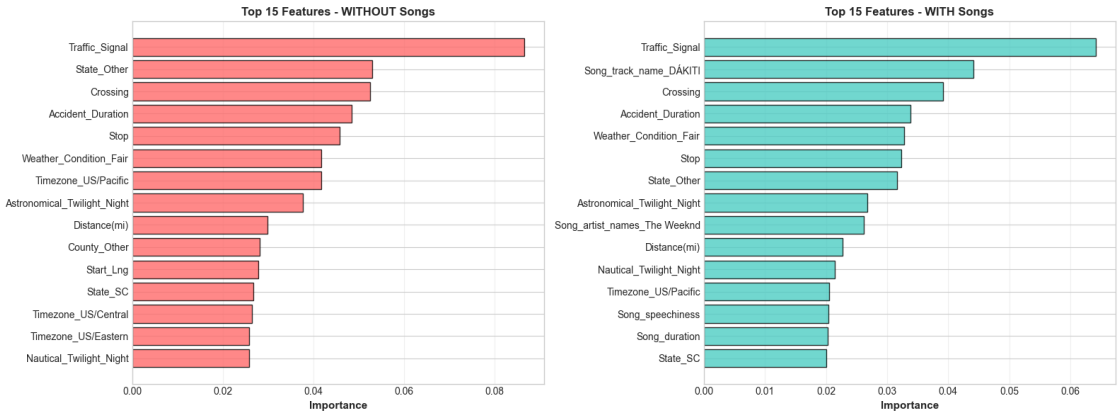

Figure 4: Feature Importance Analysis resulting from the XGBoost (Optimized), showing features as Song_track_name_DÁKITI, Song_artist_names_The Weeknd, Song_speechiness, and Song_duration.

8 Conclusion and Discussion

In this study, we successfully integrated disparate datasets to analyze the novel relationship between music consumption trends and traffic accident severity. Our experiments on over 4.1 million accidents between 2017-2021 confirmed that Spotify audio features provide a significant, though marginal, improvement to severity prediction models.

Notably, music features appeared to have high feature importance within our best performing models, despite not drastically improving model performance. The slight improvement in R^2 to 0.4352 suggests music features may serve as a valid representation for unobserved variables, such as driver demographics, seasonal patterns or specific driving norms. Aggregated music features may proxy the "vibe" of a region, rather than the specific effects of listening to music while driving. For example, high-energy music might correlate with a time of year with a higher younger driving rate, indirectly predicting higher accident risks.

8.1 Limitations and Future Work

A primary limitation is the temporal and locational granularity; mapping weekly charts to second-level accident timestamps likely smooths out acute effects. During our data collection we encountered issues in sourcing high quality and publicly available music listening data. Future work could focus on obtaining granular telematics data or exploring audio signal processing to extract more nuanced features than the metadata provided by the Spotify API.
Additionally, our main analysis centers around comparing a reduced model with no music features to a full model with all music features. Future work could look more closely at which individual music features contribute most to the improved model performance.

References

[1] Brodsky, W. (2001). The effects of music tempo on simulated driving performance. *Transportation Research Part F: Traffic Psychology and Behaviour*, 4(4), 219-241.

[2] Korkmaz, H., & Erturk, M. A. (2024). Prediction of the traffic incident duration using statistical and machine-learning methods: A systematic literature review. *Techological Forecasting and Social Change*, 207.

[3] Li, R., Chen, Y. V., & Zhang, L. (2019). Effect of music tempo on long-distance driving: Which tempo is the most effective at reducing fatigue? *i-Perception*, 10(4).

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

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

[6] Talluri, K. K., Weidl, G., & Vaishnavi, K. (2025) Accident-Driven Congestion Prediction and Simulation: An Explainable Framework Using Advanced Clustering and Bayesian Networks.