The Analysis of Car Accidents in the United States

Group 8: Eden Kidane, Jacob Timoteo, Salma Ibrahim

San Jose State University

CS 133

Professor Westfall

November 30, 2025

Table of Contents

Abstract

This project uses U.S. Accidents dataset to study the factors that relate to accident severity. We begin with basic plots that look at weather, location, and time. These plots help us see clear patterns in the data and guide which features matter the most. We then build three machine learning models using Logistic Regression, Random Forest, and XGBoost. Each model uses the same pipeline for cleaning, encoding, and training. We use N-fold cross-validation to compare their performance and after reviewing the results, we select the model with the strongest Mean Cross Validation accuracy. This report explains how we prepared the data, trained the models, tested them, and reviewed the final outcomes.

Background

Problem Description

We worked with a dataset that recorded accidents in the U.S. from 2016 to 2023. It was a very large data set with over 7.7 million records, so we took a sample of 500,000 to work with. Our goal was to explore the data and answer five main questions through visualizations and machine learning. The questions focused on environmental factors, location-based patterns, time trends, and the role of traffic and weather in accident likelihood.

To study the data, we used plots that showed weather, lighting, time of day, and state-level accident rates. We also looked at long-term trends to see how accident counts changed across seasons and years. These visual tools helped us identify patterns that guided the features we used later in our models. For version control, we used GitHub so each member could contribute in a clean and organized way. We used libraries such as plotly.express to build an interactive heat map that displayed accident counts across states and years. With scikit-learn, we prepared the data and trained models that predict accident severity using features such as weather, time, and location.

Data Exploration Questions

1. What are the most common environmental factors (weather, lighting, time of day) associated with severe crashes?

   Plot: Bar Plots

   Summary: Most severe crashes also occurred when visibility was moderate at around 5-10 miles, meaning that even when visibility reduces just a little bit from the 10-20 mile range down to the 5-10 mile range, visibility can contribute to more serious accidents. Daytime crashes were more frequent than nighttime crashes, and that is probably due to more cars on the road during the day. The results show that severe crashes mostly happen under common environmental conditions and not extreme weather, visibility(mi), or at night, as most would assume.
2. Which locations or states experience the highest accident frequency per population or per vehicle miles?

   Plot: choropleth, bar chart

   Summary: Used a second dataset to get vehicle miles traveled and merge with our current dataset. We found that several states displayed a high number of crashes relative to how many people drive. South Carolina had the highest accident rate per million VMT. This measure differs from the raw accident counts because it adjusts for total driving activity, revealing which states are truly higher risk based on a per-mile basis. This result shows that location can play an important role in accident likelihood.
3. Can we predict the severity of an accident based on time and weather conditions?

   Plot: Heatmap, Pairwise

   Summary: Severe accidents appear more common during dangerous weather such as rain, snow, and storms, while clear and cloudy conditions tend to produce lower-severity outcomes. Together, these visualizations show that time and weather contain meaningful patterns that machine-learning models can use to predict accident severity.
4. How have accident rates changed over time, and what temporal patterns exist (daily, weekly, seasonal)?

   Plot: monthly heatmaps (interactive)

Summary: There seems to be more crashes in November and December as there are a lot of drivers during the holidays. A period during 2020 shows a dip in crashes as not a lot of people were out driving during that time.

5. Are there relationships between traffic density, weather conditions, and accident likelihood?

   Plot: stacked bar plot, bar plot, heatmap

   Summary: Visualizations show patterns about how traffic density and weather jointly influence accident risk. Rush hour consistently increases accident frequency, showing that high traffic volume is a major contributor regardless of environmental factors. At the same time, adverse weather such as rain, snow, or storms, significantly increases the likelihood of severe accidents even when traffic levels are comparable. These findings support the inclusion of both time features (Hour and Day of Week) and weather features (Weather_Simple, Visibility, Precipitation, Humidity, and Wind Speed) in our machine learning models.

Overall organization and design of the program

Train/Test Split

The data was initially split to make sure that the final model provides an unbiased estimate of the performance on new data (the test data). The data set was split into training and test sets using a ratio of 80% for training and 20% for testing (test_size=0.2). The split used the stratify=df.Severity or stratify=y (y == target, target ==Severity) parameter during the split. The purpose of stratifying the split was to maintain the distribution of the target class across both the training and test sets, because the data set is imbalanced and stratifying the data set is a method used to avoid bias in performance metrics. After splitting the training and testing data and dropping rows with missing values in the chosen features the training data size was 350,888 rows and the test data size was 87,662 rows.

Data Preparation and Pipeline

To prevent data leakage, all preprocessing steps were performed after the train/test split. NA values were dropped after the split, as there was not a significant amount of missing values. A Pipeline, along with a ColumnTransformer was used so that numerical and categorical features were processed correctly and consistently across all models

Numerical features included were: 'Visibility(mi)', 'Precipitation(in)', 'Humidity(%)', 'Wind_Speed(mph)', 'Hour', 'Day_of_Week', 'Month', 'Lat', 'Lng'

Used SimpleImputer(strategy="median") to fill any possible missing values while preserving feature distribution. StandardScaler was also used to standardize all numeric variables to a zero mean and unit variance, to prevent features with large numeric ranges from dominating model training.

Categorical features included were: 'Weather_Simple', 'Sunrise_Sunset'

These were transformed using OneHotEncoder, which converts each categorical variable into binary columns, which lets the machine learning models interpret the categorical features numerically, as it cannot handle categorical input as is.A Column transformer combined the numerical and categorical transformations into a unified pipeline. The preprocessing pipeline was used directly in each machine learning model, making sure all had the same exact treatment of features during cross-validation and final testing.

ML Models Used

The target variable (Severity) is a discrete class label, all selected models were supervised classification algorithms. We trained and compared three models:

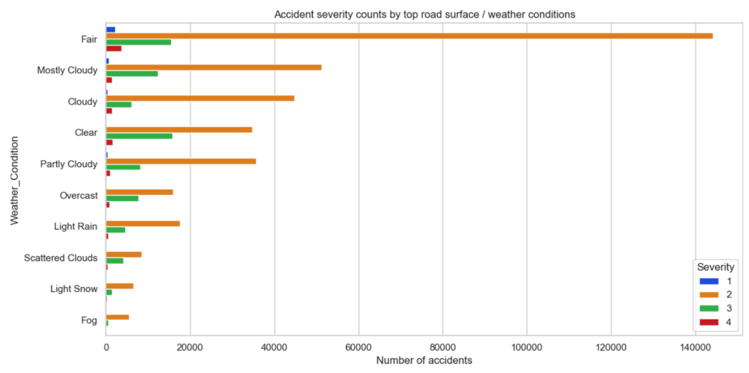
Model	Type	Reason
Logistic Regression	Classification	Baseline linear classifier, interpretable and efficient.
Random Forest Classifier	Classification	Uses ensemble decision trees to improve stability and reduce overfitting
XGBoost Classifier	Classification	Powerful gradient boosted, well suited for multi class prediction

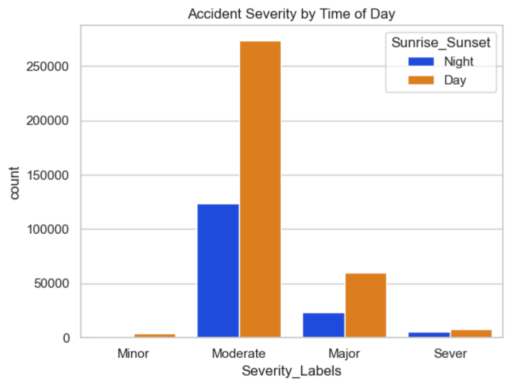
All models were evaluated using N-fold cross-validation:

scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy', n_jobs=-1)

Results

Graphs for initial data exploration





Description of Test Data and Strategy

For our test data, we did a 80/20 train test split. Since our target (severity) had some class distribution imbalance, we stratified our data to ensure there were no biases or imbalance when it came to testing. We used a total of 11 features, 2 categories (Weather_Simple, Sunrise_Sunset), and 9 Numerical (Lat, Lng, Hour, Day_of_Week, Month, Visibility, Precipitation, Humidity, Windspeed). We made sure to handle nulls properly by dropping rows with missing values after the split to prevent data leakage. In our preprocessing pipeline we used OneHotEncoder for categorial features and Simple imputer for numerical features. Then to ensure stable estimates, we used a stratified k-fold (cv = 5) on the training set.

Choosing the Best Model

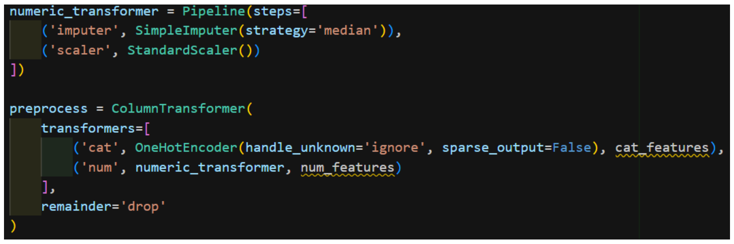
Model	Mean Cross-Validation Accuracy
Logistic Regression	0.7770
Random Forest Classifier	0.7773
XGBoost Classifier	**0.7918**

Final Test Results (Best Model)

XGBoost had slightly higher CV accuracy than Logistic Regression and RandomForest, with the top accuracy being at 79%. XGBoost is a gradient boosting algorithm, which is why it is able to achieve higher accuracy than the other models. Although this was still lower than what we had hoped for, it shows that car accidents are much more random and depend on many more variables than the ones included in our data set. It does show that there is a correlation between variables such as date, time, weather, and location, but overall it is difficult to predict the severity of an accident based on these features alone.

Code Snippets

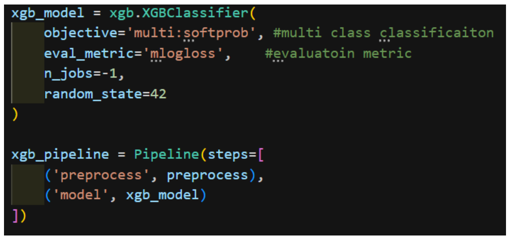
Preprocessing pipeline handling numerical and categorical data XGBoost model and preprocessing



```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

preprocess = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), cat_features),
        ('num', numeric_transformer, num_features)
    ],
    remainder='drop'
)
```

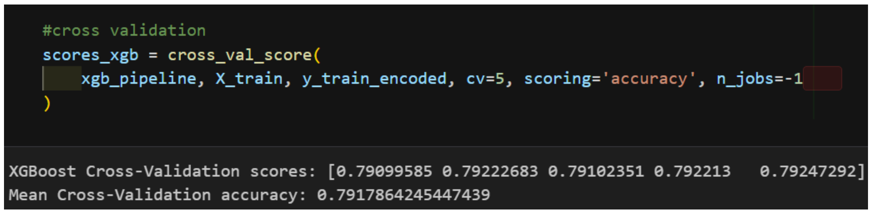
XGBoost model and preprocessing pipline



```
xgb_model = xgb.XGBClassifier(
    objective='multi:softprob', #multi class classificaiton
    eval_metric='mlogloss',    #evaluatoin metric
    n_jobs=-1,
    random_state=42
)

xgb_pipeline = Pipeline(steps=[
    ('preprocess', preprocess),
    ('model', xgb_model)
])
```

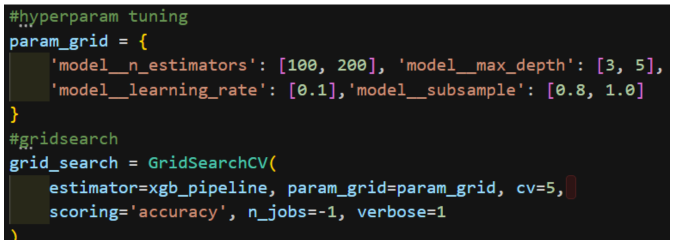
XGBoost cross-validation and output of 5 scores and their mean



```
#cross validation
scores_xgb = cross_val_score(
    xgb_pipeline, X_train, y_train_encoded, cv=5, scoring='accuracy', n_jobs=-1
)
```

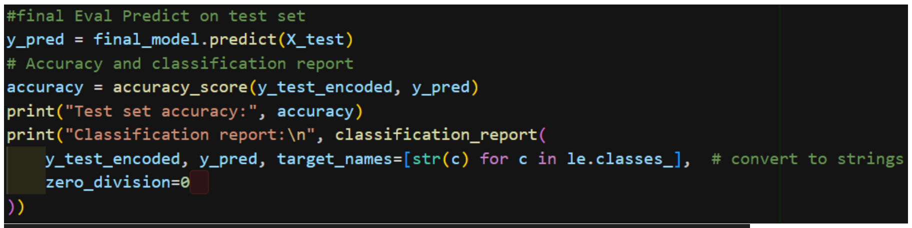
```
XGBoost Cross-Validation scores: [0.79099585 0.79222683 0.79102351 0.792213   0.79247292]
Mean Cross-Validation accuracy: 0.7917864245447439
```

Grid Search for Hyperparameter Tuning

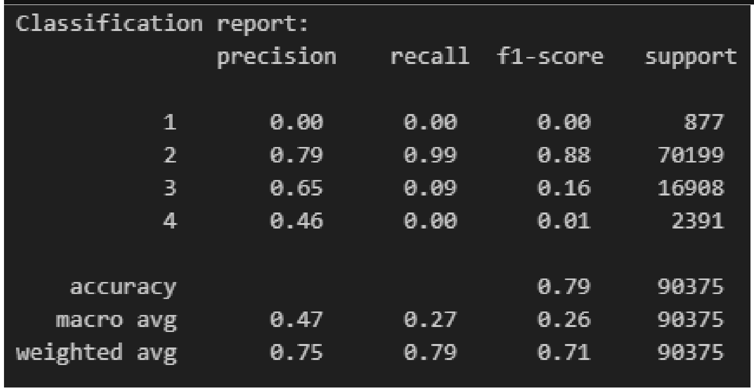


```
#hyperparam tuning
param_grid = {
    'model__n_estimators': [100, 200], 'model__max_depth': [3, 5],
    'model__learning_rate': [0.1],'model__subsample': [0.8, 1.0]
}
#gridsearch
grid_search = GridSearchCV(
    estimator=xgb_pipeline, param_grid=param_grid, cv=5,
    scoring='accuracy', n_jobs=-1, verbose=1
)
```

Final Evaluation and Classification Report and its Output (1= Minor, 2= Moderate, 3= Major, 4= Severe)



```
#final Eval Predict on test set
y_pred = final_model.predict(X_test)
# Accuracy and classification report
accuracy = accuracy_score(y_test_encoded, y_pred)
print("Test set accuracy:", accuracy)
print("Classification report:\n", classification_report(
    y_test_encoded, y_pred, target_names=[str(c) for c in le.classes_],  # convert to strings
    zero_division=0
))
```



```
Classification report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00       877
           2       0.79      0.99      0.88     70199
           3       0.65      0.09      0.16     16908
           4       0.46      0.00      0.01      2391

    accuracy                           0.79     90375
   macro avg       0.47      0.27      0.26     90375
weighted avg       0.75      0.79      0.71     90375
```

Instructions on how to run the code.

1. Download and unzip the project folder: The zip file will extract into a folder named:

   “term-project-group8-main”

2. Open the folder in a Python environment

   You may use:

   - Jupyter Notebook or JupyterLab/
   - VS Code with python + Jupyter extensions

3. Make sure you’re using Python 3.8 or above
4. Install dependencies by putting this command in the terminal
   - pip install -r requirements.txt
5. First, open and run Data/DataExploration.ipynb
   - This will clean and generate a csv file
6. Place us_accidents_sample_500k_clean.csv into the Data/ folder
7. Run the notebooks in Scripts/ in order

References

https://data.transportation.gov/Roadways-and-Bridges/Vehicle-Miles-of-Travel-by-Functional-System-and-S/nps9-3pm2/about_data

https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

https://data.transportation.gov/