Mechatronics and Intelligent Transportation Systems
https://www.acadlore.com/journals/MITS

# Machine Learning for Road Accident Severity Prediction

Koteswararao Kodepogu[1*], Vijaya Bharathi Manjeti[2], Atchutha Bhavani Siriki[3]

[1] Department of CSE, PVP Siddhartha Institute of technology, 520007 Vijayawada, India
[2] Department of CSE, GITAM School of Technology, 530045 Visakhapatnam, India
[3] Department of CSE, ANITS, 531162 Visakhapatnam, India

[*] Correspondence: Koteswararao Kodepogu (kkrao@pvpsiddhartha.ac.in)

**Received:** 10-13-2023 **Revised:** 11-16-2023 **Accepted:** 11-23-2023

**Citation:** K. Kodepogu, V. B. Manjeti, and A. B. Siriki, "Machine learning for road accident severity prediction," *Mechatron. Intell Transp. Syst.*, vol. 2, no. 4, pp. 211–226, 2023. https://doi.org/10.56578/mits020403.

**Abstract:** In the realm of road safety management, the development of predictive models to estimate the severity of road accidents is paramount. This study focuses on the multifaceted nature of factors influencing accident severity, encompassing both vehicular attributes such as speed and size, and road characteristics like design and traffic volume. Additionally, the impact of variables, including driver demographics, experience, and external conditions such as weather, are considered. Recent advancements in data analysis and machine learning (ML) techniques have directed attention toward their application in predicting traffic accident severity. Unlike traditional statistical methods, ML models are adept at capturing complex variable interactions, thereby offering enhanced prediction accuracy. However, the efficacy of these models is intrinsically tied to the quality and comprehensiveness of the utilized data. This research underscores the imperative of uniform data collection and reporting methodologies. Through a meticulous analysis of existing literature, the paper delineates the foundational concepts, theoretical frameworks, and data sources prevalent in the field. The findings advocate for the continuous development and refinement of sophisticated models, aiming to diminish the frequency and gravity of road accidents. Such efforts contribute significantly to the enhancement of traffic control and public safety measures.

**Keywords:** ML; Road safety; Traffic accident analysis; Public safety; Traffic management

## 1 Introduction

Road traffic accidents, a prevalent public health concern, are not only sources of human distress but also exert substantial economic and societal impacts. The consequences of these accidents extend beyond the immediate medical and rehabilitation costs, encompassing property damage, productivity losses, and increased insurance premiums. The long-term effects on victims, families, and communities further exacerbate this burden. The precise prediction of accident severity thus emerges as a pivotal factor in mitigating the societal impacts of road traffic incidents. Leveraging ML and data analysis techniques enables the exploration of diverse factors contributing to traffic accidents, such as weather conditions, traffic patterns, road design, and driver behavior. These factors are integral to the development of accurate models for predicting accident severity, which, in turn, aid emergency services in delivering prompt and effective responses. A notable benefit of such accurate predictions is the enhanced management of emergency resources, allowing for prioritization based on the severity of incidents. This leads to swifter response times and improved outcomes for those involved in accidents. Additionally, precise predictions can inform transportation management strategies and the design of safer road networks, ultimately reducing the likelihood and severity of future accidents.

The challenges in applying ML and data analysis for accident severity prediction include the availability and quality of data, model complexity, and ethical concerns regarding the use of private information. Despite these challenges, this field holds substantial implications for traffic and public safety management. Current models, incorporating human behavior, road types, weather conditions, and vehicle types, are yet to reach full effectiveness due to the dynamic and multifactorial nature of traffic accidents. Additionally, managing large data volumes can hinder decision-making processes, affecting emergency response efficiency. Addressing these challenges requires the development of a reliable and accurate model that accounts for all relevant factors and can process vast data amounts, adapt to changing conditions, and provide real-time insights. This solution could significantly reduce the number of accidents, injuries, fatalities, and enhance emergency response efficiency. The overarching goal of road accident severity prediction is the early and precise determination of an accident's severity, facilitating prompt and efficient deployment of emergency response teams. This initiative aims to minimize fatalities, injury severity, and the economic burden of these incidents. Accurate severity assessment aids policymakers in identifying high-risk areas and implementing appropriate safety measures. Moreover, this prediction can foster the development of preventive strategies, such as enhanced traffic management, road design improvements, and public awareness campaigns, to reduce accident likelihood. The ultimate objective is to elevate emergency response efficiency, decrease the frequency of collisions, and enhance overall traffic safety.

## 2 Literature Review

In the domain of road traffic accident severity prediction, various studies have contributed significantly, employing ML techniques to enhance predictive accuracy. Paul et al. [1] introduced a multiclass model that amalgamates accident prediction with severity assessment, employing a suite of algorithms including Decision Tree, Random Forest, Multilayer Perception, and Categorical Naïve Bayes. This model aims to not only predict road collisions but also to delineate their severity, thus offering a comprehensive approach to traffic safety.

Mallahi et al. [2] focused on predicting traffic accident severity, which is crucial for optimizing emergency logistical responses in urban areas. Their study utilized ML algorithms such as Random Forest, Support Vector Machine (SVM), and Artificial Neural Networks (ANN), classifying traffic accident severity into three categories: Pedestrian, Vehicle or Pillion Passenger, and Driver or Rider. The research employed the TRAFFIC ACCIDENTS_2019_LEEDS dataset from the Road Safety department of Transport for this classification.

Yang et al. [3] proposed an enhanced model for predicting traffic accident severity based on the random forest algorithm. Their innovative approach extended existing models by incorporating four critical accident characteristics: the accident site, form, road information, and driving speed. Their findings indicated that the random forest algorithm outperformed other neural network models like BP, SVM, and RBF in terms of prediction accuracy. Notably, the collision pattern emerged as a pivotal factor in determining the severity of traffic accidents.

Scholars including Sowdagur et al. [4], Labib et al. [5] and Jadhav et al. [6] addressed the global concern of traffic accident causes and severity, with a specific focus on Mauritius. Employing ML strategies, particularly ANN, they experimented with various configurations of multilayer perceptron (MLP). Through systematic manual tuning, optimal hyperparameters were determined, including a limit of 10,000 iterations, a learning rate of 0.1, and specific neuron counts in the hidden layers. The Rectified Linear Unit (RELU) was employed as the activation function. The study effectively mitigated overfitting and underfitting issues using stratified 10-fold cross-validation.

## 3 Proposed Methodology

The raw data of the dataset must undergo transformation into a format conducive to machine processing as part of the data analysis and preprocessing procedure. This transformation leverages both statistical and logical methods to describe, present, compile, rate, and analyze the data. The workflow is illustrated in Figure 1.

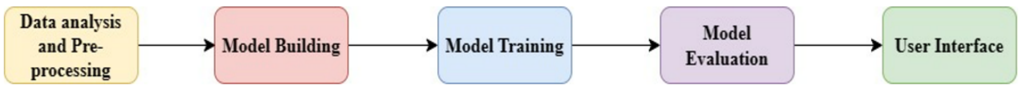


**Figure 1.** Flow of the work

**Table 1.** Types of attributes in dataset

| Names of Attributes | Few Categories of Attributes |
|---|---|
| Type of vehicle | Automobile, Public transport, Lorry, Taxi, Pick up to 10Q, Station wagon, Ridden horse, Other, Bajaj, Turbo, Two Wheeler, Special vehicle. |
| Road surface type | Asphalt roads, Earth roads, Asphalt roads with some distress, Gravel roads, and Others. |
| Road surface conditions | Dry, Wet or damp, Snow, Flood over 3cm. deep |
| Light conditions | Daylight, Darkness - lights lit, Darkness - no lighting |
| Weather conditions | Normal, Raining, Cloudy, Snow, Windy |
| Sex of driver | Male, Female, Unknown |
| Age band of driver | 18-30, 31-50, Under 18, Over 51, Unknown |
| Driving experience | 1-2 yr, Above 10 yr, 5-10 yr, 2-5 yr, No licence, Below 1 yr, Unknown |
| Accident severity | Slight injury, Serious injury, Fatal injury |

The dataset employed in this study comprises textual data associated with nine variables, each delineating different factors that impact the severity of traffic accidents (see Table 1). Among these variables is 'Accident Severity', which serves as the primary attribute of interest. The dataset encompasses 12,316 records, each providing a detailed account of various sub-attributes that contribute to predicting the degree of accident severity. This comprehensive analysis is not only critical in a theoretical context but also holds the potential to save lives and enhance the efficiency of emergency response services in real-world scenarios.

Following the data analysis and preprocessing stages, several algorithms were selected for the development of a ML model. This approach facilitates a comparative analysis of various models. The dataset was utilized to train the model, enabling an evaluation of the model's accuracy in predicting correct outcomes. The data was split into two segments: 80% for training and 20% for testing purposes. Consequently, a training dataset consisting of 9,852 records was used to train the algorithm.

Model evaluation is a critical phase in which a range of metrics are employed to assess a ML model's performance, highlighting its strengths and limitations. It is imperative to evaluate the efficacy of the model early in the research process. Model evaluation not only aids in monitoring the model's performance but also informs subsequent improvements. For the assessment of the model, various visualization tools such as line charts and bar graphs were employed. Additionally, a confusion matrix was used to illustrate the algorithm's performance in categorizing predicted accident severity levels [7].

Upon the completion of the model's development, training, and testing phases, a webpage was designed using Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). The Flask framework, supplemented by the Pickle module, was utilized to integrate the ML model with the website, facilitating an interactive user experience.

## 4 ML

ML enables autonomous learning from data and previous experiences, identifying patterns and making predictions with minimal human intervention. It employs data and algorithms to emulate human learning processes, thereby enhancing accuracy over time. A key aspect of ML is its ability to execute tasks without explicit programming. It involves the analysis of algorithms and structures within datasets. Attributes of these datasets are utilized as inputs during the training and testing phases of algorithms. The efficacy of a ML algorithm is often contingent upon the accuracy of the input-data representation. Studies have demonstrated that optimal data representation significantly enhances performance, as compared to sub-optimal data representation. The ubiquity of ML in research and its application across various fields, including image classification, is well documented [5].

### 4.1 HTML

HTML, an acronym for Hypertext Markup Language, is the foundational language for creating web pages and web content. It comprises a series of elements or tags, which dictate the structure, content, and layout of a webpage. HTML tags typically include an opening tag, optionally a closing tag, and the content of the tag, all enclosed within angle brackets. These tags mark the beginning and end of an HTML element, with self-closing elements not requiring an ending tag. HTML components such as headings, paragraphs, lists, links, images, tables, forms, and more, define various aspects of a webpage. They also facilitate the application of CSS for styling. Mastery in crafting and modifying HTML documents is integral to developing engaging and functional webpages, making it a fundamental skill in web development.

### 4.2 CSS

CSS, or Cascading Style Sheets, is a styling language used to define the presentation of documents written in HTML or XML. Its primary function is to separate the presentation aspect of a document from its content, thereby simplifying the process of modifying the appearance of multiple web pages simultaneously. In CSS, selectors are employed to apply styling rules to specific HTML elements. These rules determine the style attributes to be applied within the chosen scope. For example, CSS can dictate that all paragraph elements should have a line height of 1.5em and a font size of 16 pixels. It allows for the styling of various document elements, including borders, backgrounds, images, and text. The extensive range of style values and properties available in CSS empowers designers to create intricate and visually appealing layouts. As a vital tool for web designers, CSS is often utilized in conjunction with HTML and JavaScript to develop modern, responsive websites, enabling the creation of visually compelling web pages while optimizing performance.

### 4.3 Flask Framework

Flask is a widely utilized web application framework within the Python community. It is characterized as a lightweight and flexible framework, providing developers with essential tools for rapid web application development. Being a micro-framework, Flask incorporates only the fundamental features required for web application creation.

This minimalistic approach renders it user-friendly and easy to learn, allowing developers to tailor their applications by incorporating only necessary features. Flask's key attributes are outlined in Table 2.

**Table 2.** Attributes

| Names of Attributes | Details |
|---|---|
| Routing | Flask offers an easy-to-use interface for specifying URL routes that handle HTTP queries. |
| Templates | Jinja2 templates, which offer a potent way to build dynamic HTML pages, are supported by Flask. |
| Supports for database | Flask works well with various databases, including NoSQL databases like MongoDB and SQL databases like SQLite, MySQL, and PostgreSQL. |
| Testing and Troubleshooting | Flask has integrated debugging and testing features that assist programmers in finding and fixing bugs in their code. |
| Extensions | Flask has a robust community of add-ons supporting features like RESTful API development, caching, and authentication. |

## 5 Algorithms

### 5.1 Decision Tree

The decision tree, a favored technique in ML, is extensively employed for both classification and regression tasks. As a supervised learning method, it partitions the dataset into smaller subsets based on the input features, subsequently predicting values for the target variable. The decision tree algorithm recursively utilizes the feature with the highest information gain for splitting the dataset. Information gain is defined as the reduction in entropy of the dataset achieved by partitioning it based on a specific attribute. This process continues until a tree-like structure is formed, with each leaf node representing a prediction for the target variable. Decision trees are notable for their interpretability, elucidating the relationships between input features and the target variable. However, susceptibility to overfitting is a concern if the tree grows overly complex. Methods such as pruning and ensemble techniques, including random forests, are employed to mitigate this issue [8, 9].

### 5.2 Random Forest

Random Forest, an ensemble ML approach, is renowned for addressing classification and regression problems. It enhances precision and reliability by combining the predictions of multiple decision trees. Each tree in the Random Forest is trained using a randomly selected subset of the training data and constructed using a random assortment of input features. This randomization reduces the correlation between trees and diversifies their predictions. For predictions, each tree contributes its output, with the collective output being averaged (for classification tasks) or voted upon (for regression tasks). Compared to a single decision tree, Random Forest offers several advantages. Its ensemble nature reduces the likelihood of overfitting and it handles noisy data and missing values more effectively. Moreover, Random Forest is capable of ranking input features based on their contribution to the model's accuracy [10, 11].

### 5.3 K-Nearest Neighbor (KNN)

KNN, a non-parametric and supervised algorithm, is utilized for both classification and regression tasks. Its simplicity belies its effectiveness in solving a range of problems. KNN identifies the K nearest neighbors to a query point based on a chosen distance measure (such as Euclidean or Manhattan distance) in a high-dimensional space. The class or value of the query point is then determined by the majority class or the average of the values of these K neighbors. K, a crucial hyperparameter, is typically optimized through cross-validation. While larger K values may lead to over-smoothing, smaller values can make the model sensitive to noise and outliers. Although KNN is straightforward, its application in large datasets can be computationally demanding, necessitating careful choice of the distance measure and data preprocessing [12].

Conversely, smaller values of K can render the model more vulnerable to noise and outliers. While KNN is straightforward and easy to understand, its application can be computationally demanding, particularly with large datasets. Selecting the optimal distance measure and thorough preprocessing of the data are essential steps for the effective implementation of the KNN algorithm.

### 5.4 Naïve Bayes

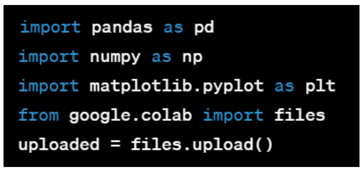
Naïve Bayes, a widely implemented classification method, is grounded in Bayes' theorem. This theorem postulates that the probability of a hypothesis given evidence is proportional to the likelihood of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis. Naïve Bayes is utilized in classification to predict an observation's class based on its features, operating under the assumption that these features are conditionally independent of the class. It calculates the probability of each class given the features' values, selecting the class with the highest probability. This calculation involves determining the prior probabilities of each class and the conditional probabilities of each feature given each class, using a training dataset. Variants of Naïve Bayes, such as Gaussian, Multinomial, and Bernoulli, cater to different data types and feature distribution assumptions. Although efficient and straightforward, Naïve Bayes can underperform in cases where the conditional independence assumption does not hold [13–15].

### 5.5 Adaptive Boosting (Adaboost)

Adaboost, an ensemble ML technique, amalgamates multiple weak learners to form a highly accurate classifier. It iteratively trains weak learners on varying subsets of the training data, assigning higher weights to misclassified observations by previous learners. The final prediction is a weighted combination of all weak learners' outputs. Each iteration adjusts the weights of observations in the training set, increasing weights for incorrectly classified and reducing weights for correctly classified observations. This approach ensures that subsequent weak learners focus more on previously misclassified observations. Adaboost's strength lies in its ability to focus on the most informative observations, enhancing the ensemble's performance iteratively. Despite its resistance to overfitting, Adaboost is sensitive to noise and outliers, which can affect the performance of weak learners and, consequently, the overall ensemble. Additionally, Adaboost can be computationally intensive due to the repetitive training of multiple weak learners on varied data subsets [16–18].

## 6 Methodology and Solution

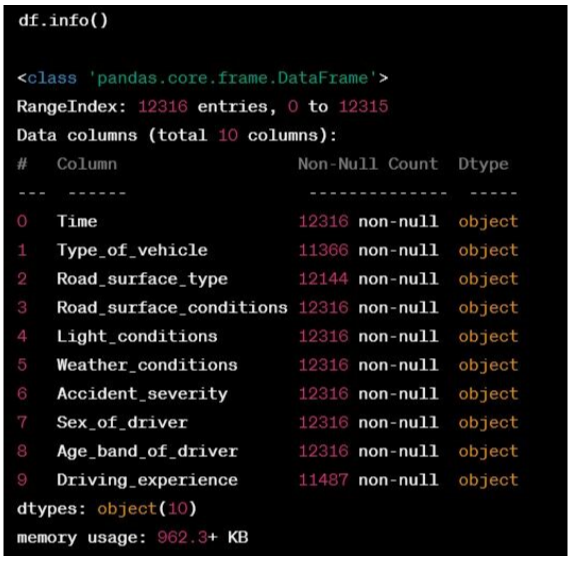
At first, all libraries to be used in this project were imported and the dataset is uploaded using the upload function of the Google colab files library.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()
```

**Figure 2.** Importing libraries

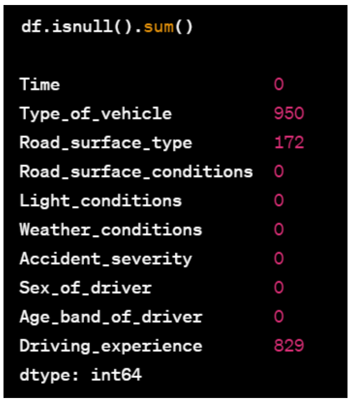
After acquiring the dataset, the dataset was read and the data information was checked to ascertain key details, including the number of attributes and the count of entries for each dataset. Figure 2 illustrates the libraries imported for this purpose.



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12316 entries, 0 to 12315
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Time                     12316 non-null  object
 1   Type_of_vehicle          11366 non-null  object
 2   Road_surface_type        12144 non-null  object
 3   Road_surface_conditions  12316 non-null  object
 4   Light_conditions         12316 non-null  object
 5   Weather_conditions       12316 non-null  object
 6   Accident_severity        12316 non-null  object
 7   Sex_of_driver            12316 non-null  object
 8   Age_band_of_driver       12316 non-null  object
 9   Driving_experience       11487 non-null  object
dtypes: object(10)
memory usage: 962.3+ KB
```

**Figure 3.** Dataset info

Then, null values within each attribute were identified using a combination of the isnull() and sum() functions. The dataset information, including these details, is depicted in Figure 3. Specifically, the attributes 'Type_of_vehicle', ‘Road_surface_type’, and ‘Driving_experience’ contained 950, 172, and 829 null values, respectively. A detailed count of these null values is presented in Figure 4.

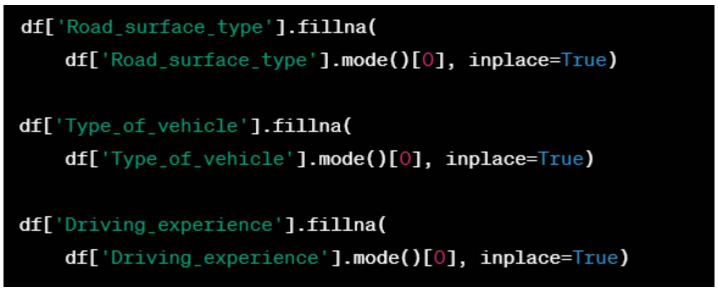


```
df.isnull().sum()

Time                       0
Type_of_vehicle            950
Road_surface_type          172
Road_surface_conditions    0
Light_conditions           0
Weather_conditions         0
Accident_severity          0
Sex_of_driver              0
Age_band_of_driver         0
Driving_experience         829
dtype: int64
```

**Figure 4.** Null values count

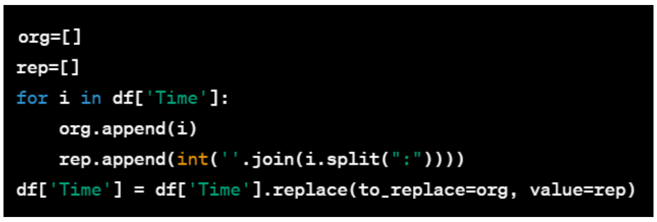
The missing values in the dataframe were addressed by substituting them with the mode of their respective columns.

In the dataset, given that the conditions were string types, the “Accident Severity” column was modified by replacing the original value codes with numbers: 1, 2, or 3, based on the respective condition. The process of replacing null values is illustrated in Figure 5. Additionally, for the “Time” attribute, the strings were converted into numerical values to facilitate processing, as demonstrated in Figure 6.



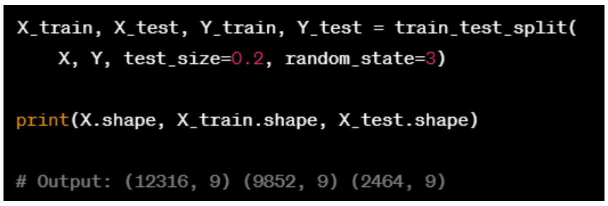
```
df['Road_surface_type'].fillna(
    df['Road_surface_type'].mode()[0], inplace=True)

df['Type_of_vehicle'].fillna(
    df['Type_of_vehicle'].mode()[0], inplace=True)

df['Driving_experience'].fillna(
    df['Driving_experience'].mode()[0], inplace=True)
```

**Figure 5.** Replacing null values



```
org=[]
rep=[]
for i in df['Time']:
    org.append(i)
    rep.append(int(''.join(i.split(":"))))
df['Time'] = df['Time'].replace(to_replace=org, value=rep)
```

**Figure 6.** Time to string conversion



```
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.2, random_state=3)

print(X.shape, X_train.shape, X_test.shape)

# Output: (12316, 9) (9852, 9) (2464, 9)
```

**Figure 7.** Splitting the dataset

The dataset, comprising 12,316 records, was divided into training and testing sets following an 80-20 split ratio. This resulted in 9,852 records for training and 2,464 records for testing, as depicted in Figure 7.

In our dataset, the most prevalent types of vehicles are Automobiles and Lorries, followed by Public Transport. Conversely, Bajaj and Turbo vehicles are the least represented. This distribution is illustrated in Figure 8.

Within our dataset, the most common road surface condition is ‘Dry’, followed by ‘Wet or Damp’. Conditions such as ‘Snow’, ‘Flood’, and surfaces ‘3cm Deep’ are less frequently observed. This categorization is depicted in Figure 9.

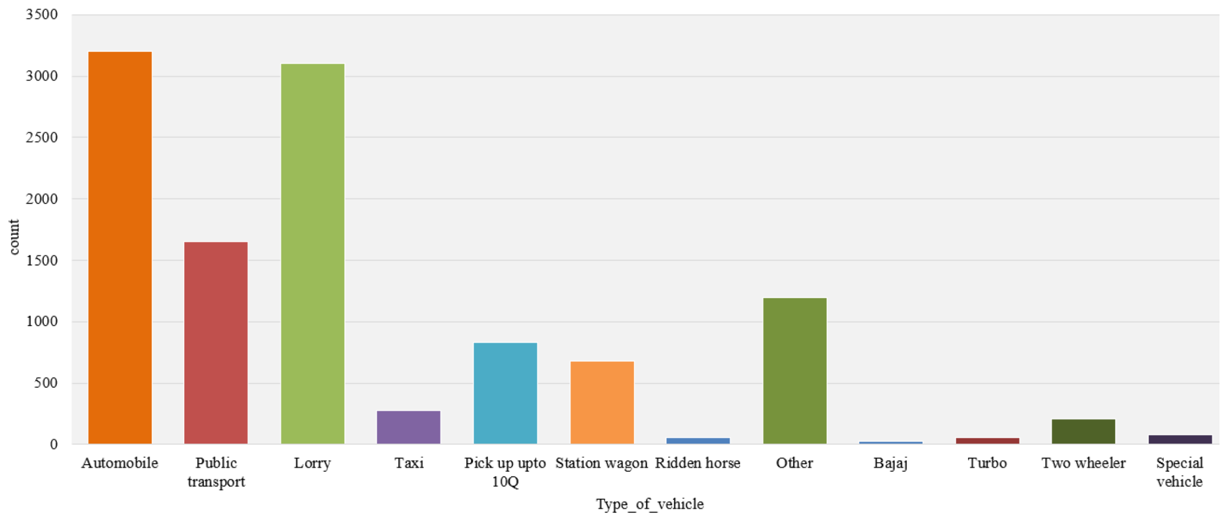


**Figure 8.** Count vs Type of vehicle

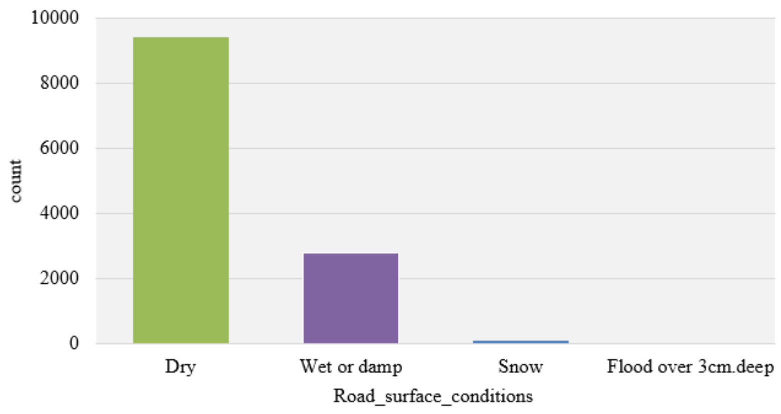


**Figure 9.** Count vs Road surface conditions

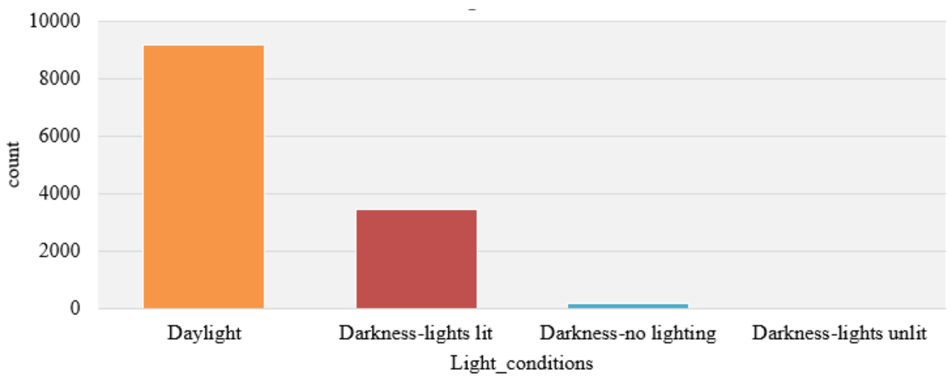


**Figure 10.** Count vs Light conditions

The graph in question, as depicted in Figure 10, indicates that the majority of the dataset’s records fall under ‘Daylight’ lighting conditions, while the ‘Darkness - No Lighting’ category has the fewest records.

In Figure 11, a significant portion of the data, approximately 10,000 records or nearly 90%, pertains to 'Normal' weather conditions. This is followed by a comparatively smaller number of records indicating 'Rain' and other weather conditions.

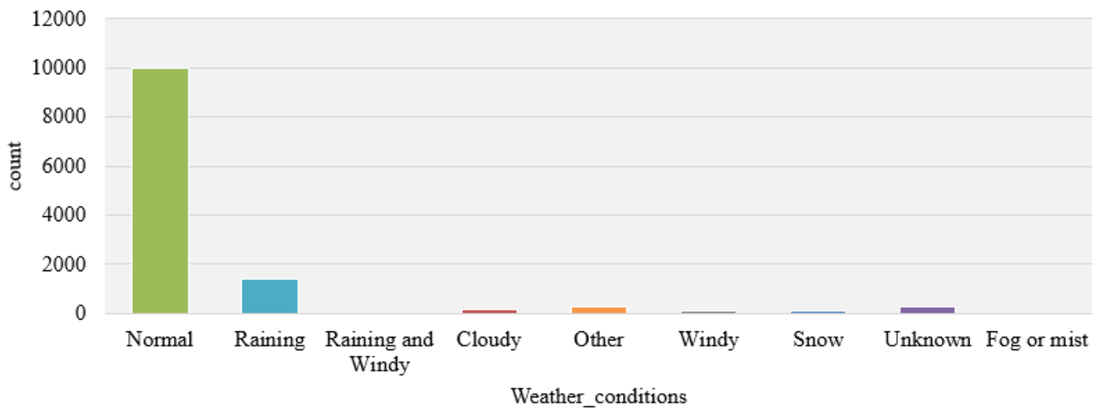


**Figure 11.** Count vs Weather conditions

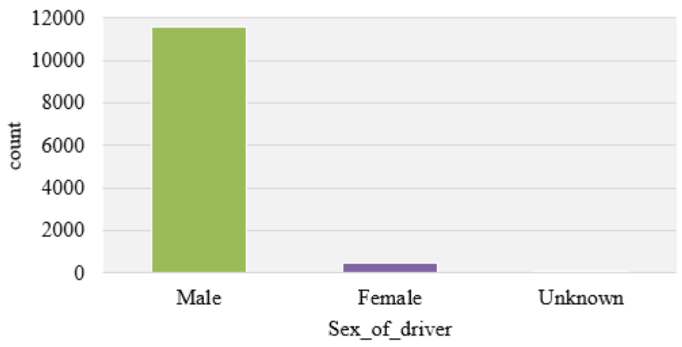


**Figure 12.** Count vs Sex_of_driver

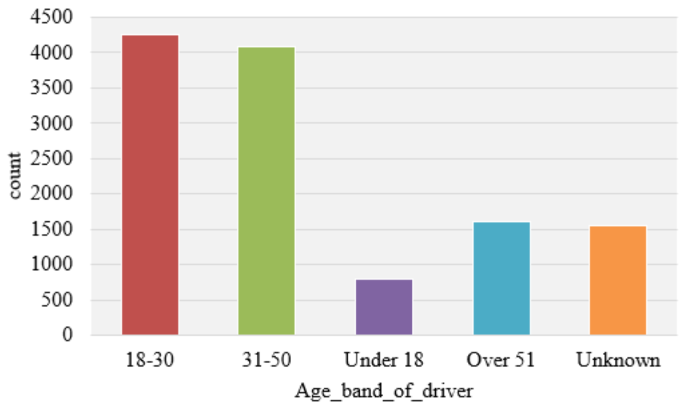


**Figure 13.** Count vs Age band of driver

Figure 12 allows us to deduce that the majority of the records in our dataset pertain to male drivers, amounting to over 11,000 instances.

The analysis of Figure 13 reveals a higher prevalence of records within the 18-50 age group. However, the presence of 'Unknown' values in the dataset necessitates either their imputation with suitable values or removal to maintain data integrity.

The data, as presented in Figure 14, indicates a higher number of individuals with 2-10 years of driving experience, followed by those with over 10 years of experience. Notably, there are no instances marked as 'Unknown' in this category, which is a positive aspect in terms of data completeness.

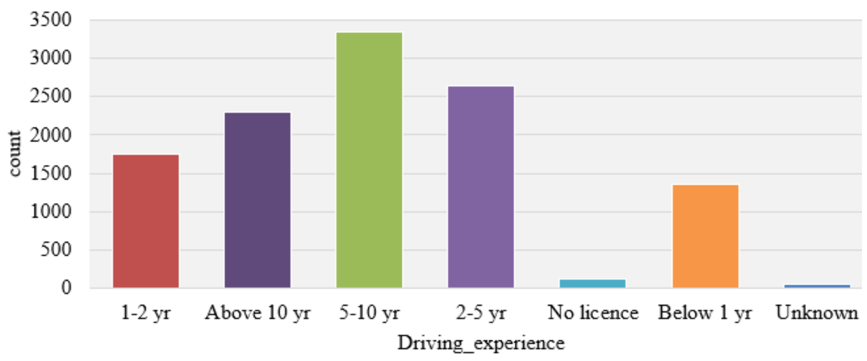


**Figure 14.** Count vs Driving experience

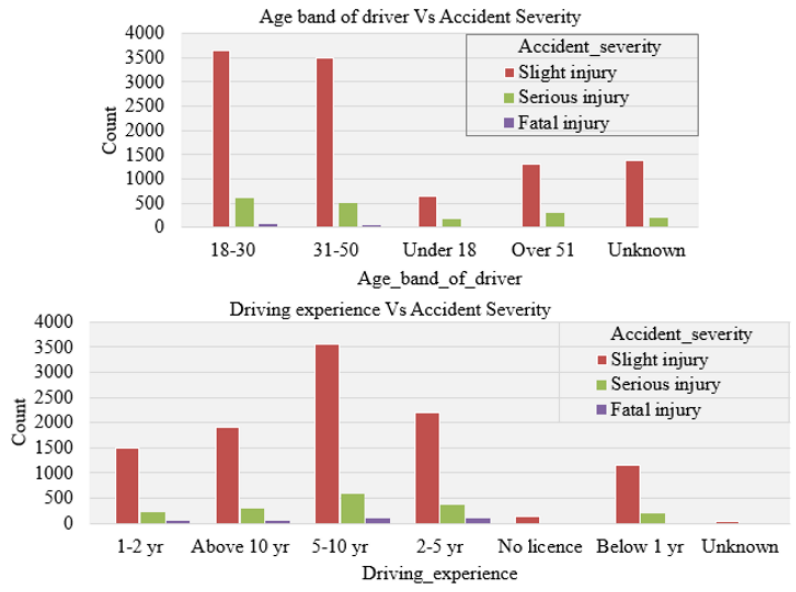


**Figure 15.** Severity vs Attributes

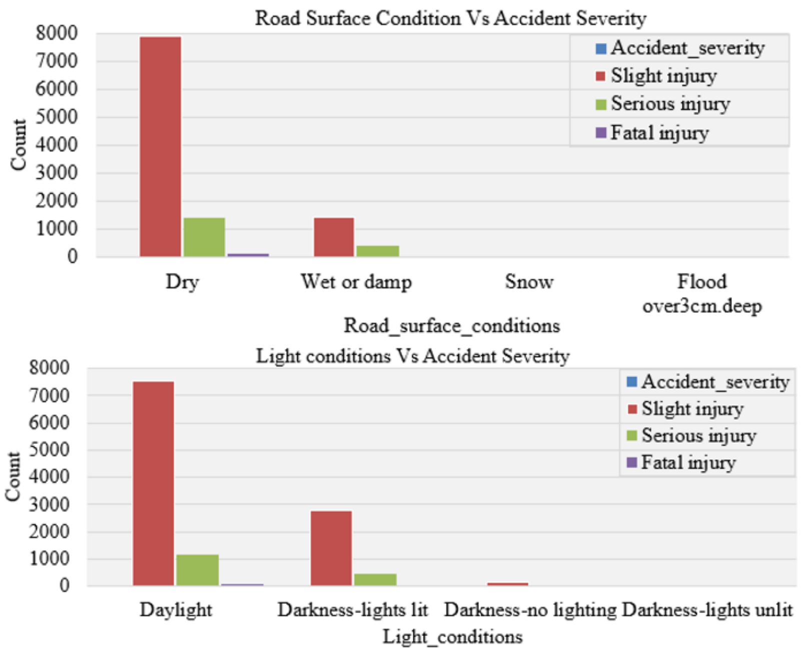


**Figure 16.** Severity vs Attributes

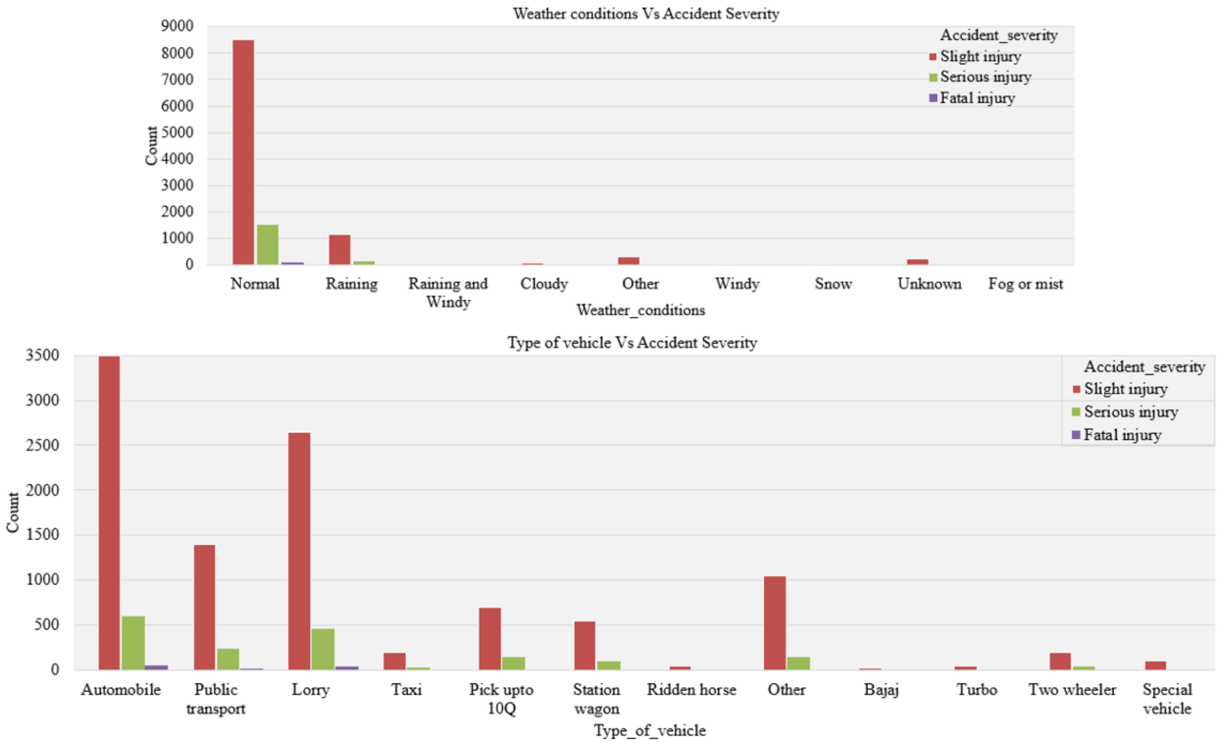


**Figure 17.** Severity vs Attributes

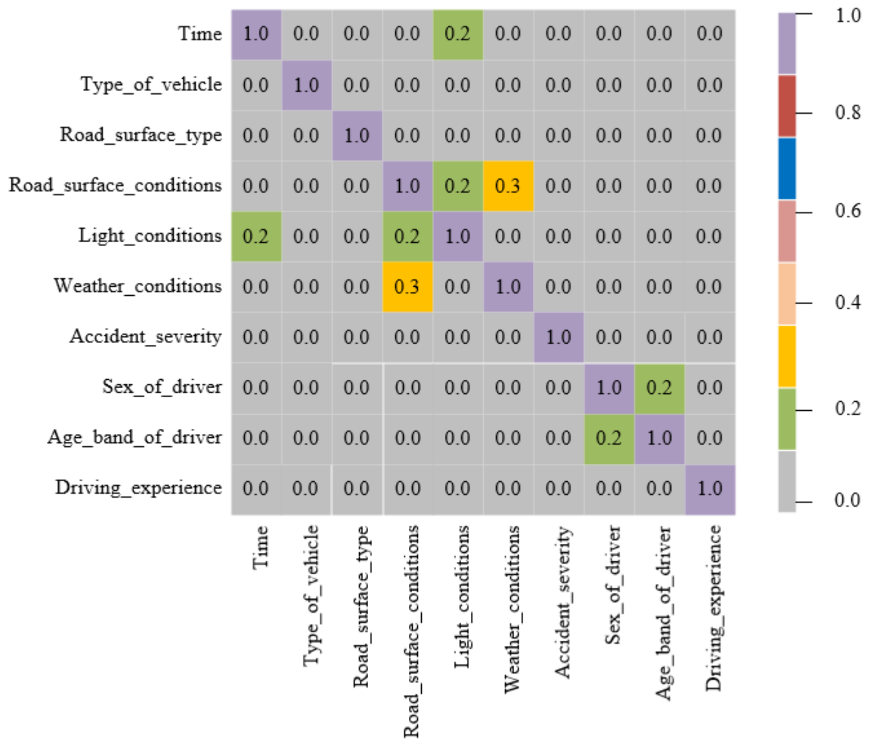


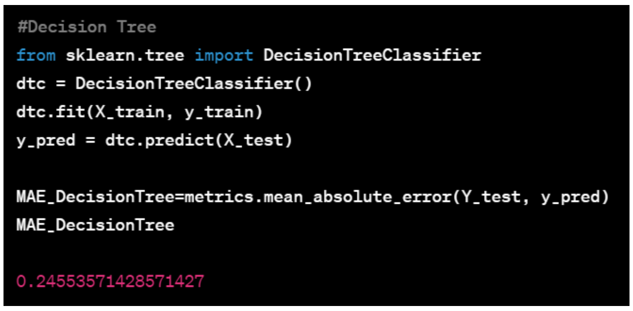
**Figure 18.** Correlation matrix

These analyses provide valuable insights into specific attributes. For instance, it was observed that serious injuries occur across all categories of driving experience when road surface conditions are dry. Additionally, a higher frequency of accidents is associated with Automobiles as the vehicle type.

Figures 15-17 elucidate the relationship between accident severity and various attributes. A notable observation is that the severity of accidents under typical weather conditions is most pronounced in the age group of 18 to 50, particularly during daylight hours.

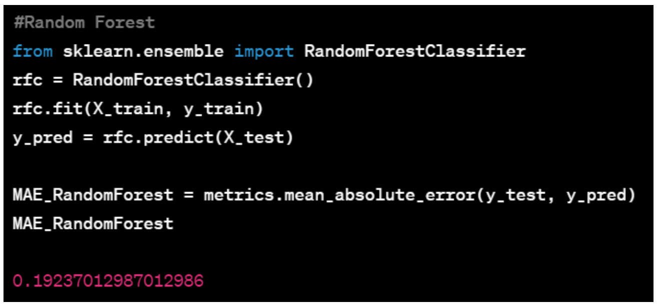
The correlation matrix, as depicted in Figure 18, indicates that all attributes in the dataset are largely independent, yet each plays a crucial role in predicting accident severity. A minor correlation was observed between attributes such as ‘Road_Surface_Conditions’, ‘Weather_Conditions’, and ‘Light_Conditions’. Additionally, the attributes ‘Age_Band_of_Driver’ and ‘Sex_of_Driver’ exhibited a slight dependency.

The implementation and outcomes of the Decision Tree, Random Forest, K-Nearest Neighbors, Gaussian Naïve Bayes, and Adaboost Classifier are depicted in Figures 19-23, respectively.



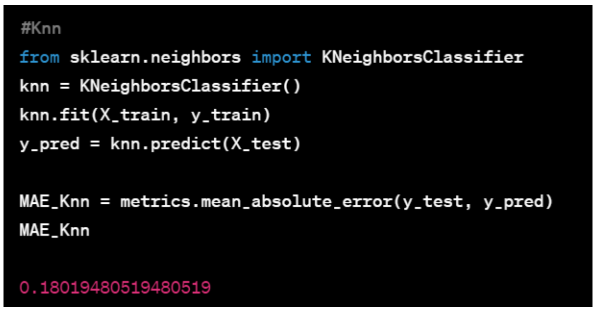
```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)

MAE_DecisionTree=metrics.mean_absolute_error(Y_test, y_pred)
MAE_DecisionTree

0.24553571428571427
```

**Figure 19.** Decision tree classifier



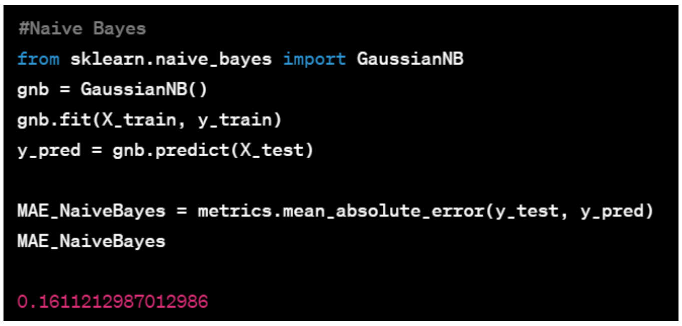
```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

MAE_RandomForest = metrics.mean_absolute_error(y_test, y_pred)
MAE_RandomForest

0.19237012987012986
```

**Figure 20.** Random forest classifier



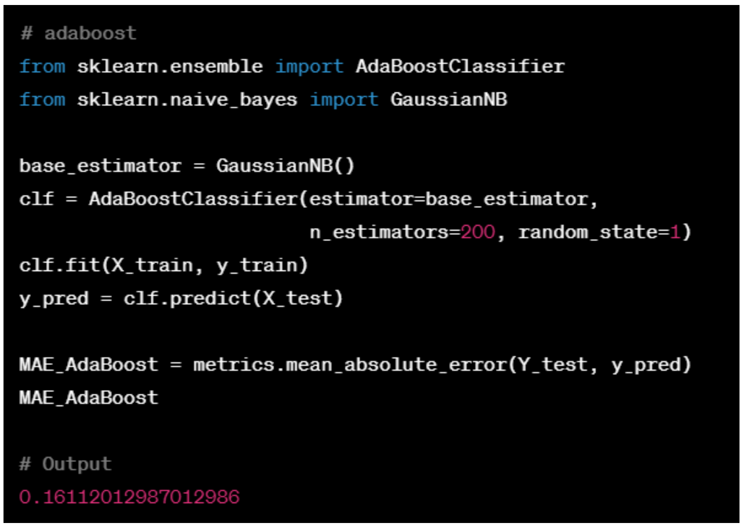
```
#Knn
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

MAE_Knn = metrics.mean_absolute_error(y_test, y_pred)
MAE_Knn

0.18019480519480519
```

**Figure 21.** K-Nearest Neighbors classifier



```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

MAE_NaiveBayes = metrics.mean_absolute_error(y_test, y_pred)
MAE_NaiveBayes

0.1611212987012986
```

**Figure 22.** Gaussian Naïve Bayes



```
# adaboost
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB

base_estimator = GaussianNB()
clf = AdaBoostClassifier(estimator=base_estimator,
                         n_estimators=200, random_state=1)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

MAE_AdaBoost = metrics.mean_absolute_error(Y_test, y_pred)
MAE_AdaBoost

# Output
0.16112012987012986
```

**Figure 23.** Adaboost classifier

Various machine learning algorithms were leveraged to train models using the designated training and testing datasets. Specifically, the Decision Tree method was first employed, and the mean absolute error of this model was calculated. This was followed by the use of the Random Forest algorithm, and the mean absolute error for this model was also computed. The K-Nearest Neighbor algorithm was then used for training, and its model's mean absolute error was determined. The Naive Bayes algorithm was next applied, with its associated model's mean absolute error calculated. Finally, the AdaBoost algorithm was used for model training, and the mean absolute error for this model was determined. Through this approach, we have trained models and computed their respective mean absolute errors for a comprehensive comparison.

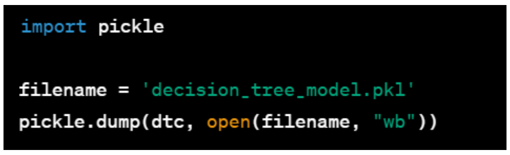
### 6.1 Flask Framework for Building User Interface

A fundamental aspect of ML involves creating and training models using diverse techniques on extensive datasets. The subsequent phase, crucial for real-world application, entails integrating these models into various software platforms. To facilitate the prediction of new data by users globally, the models must be accessible via the internet.

Flask, known for its compactness, is a web application framework ideal for developing small-scale web applications. Creating RESTful APIs with Flask and Python is a straightforward process. The basic structure of our web page will be provided in an HTML file (referred to as 'a.html'), which will be situated in the template directory. This file is then integrated with the Flask framework to generate a web page. Utilizing the Pickle package enhances efficiency, as it keeps a record of already serialized objects, preventing redundant serialization for repeated references. This feature is particularly useful for swiftly saving models, especially those that are compact with a limited number of parameters.

### 6.2 Process of ML Algorithm Connection to Webpage

The initial step involves converting the developed ML model into a Pickle file. This conversion is achieved using the 'dump()' function in the Pickle package, where the model is transformed into a '.pkl' file alongside the specified filename and write-binary mode ('wb'). This process is illustrated in Figure 24.

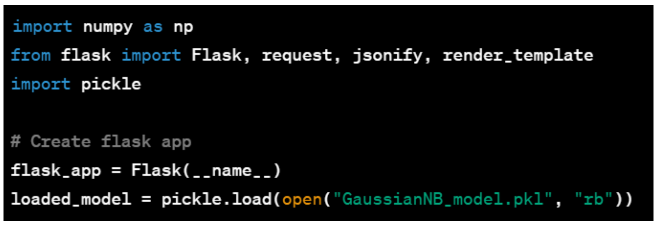


```
import pickle

filename = 'decision_tree_model.pkl'
pickle.dump(dtc, open(filename, "wb"))
```

**Figure 24.** Importing and creating object

The next step involves integrating the model into the Flask framework and establishing a connection with the HTML page, thereby enabling the web page to access the model's functionalities.

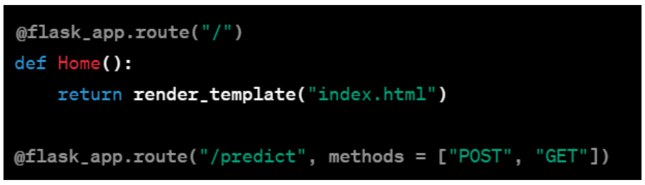
Using the Pickle package, the '.pkl' file was loaded with 'rb' (read-binary) mode as the parameter. This file was then assigned to the 'loaded_model' variable, which enabled its use as a ML model on the website. Once all components were in place, Flask was invoked to construct the web application. This process is detailed in Figure 25.



```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

# Create flask app
flask_app = Flask(__name__)
loaded_model = pickle.load(open("GaussianNB_model.pkl", "rb"))
```

**Figure 25.** Creating flask application

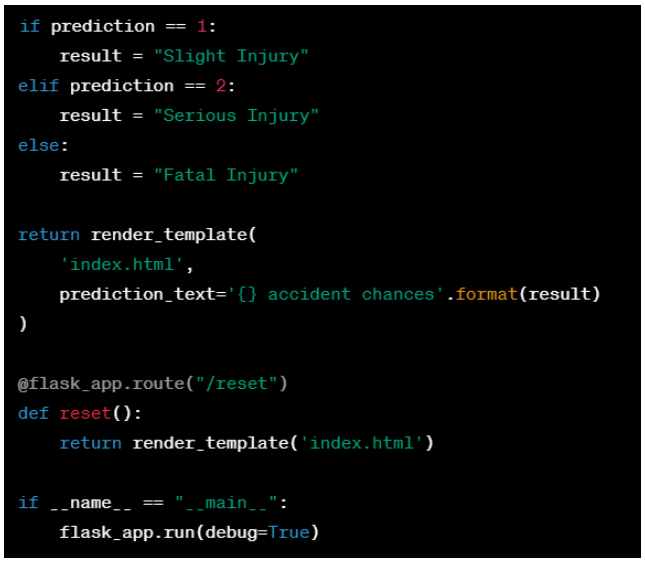
The flask_app.route() function designated the specific web page to be accessed, as indicated by its parameters, typically prefixed with a "/". Accompanying the web page URL parameter within the @flask_app.route() decorator are additional attributes, specifically the methods 'POST' and 'GET'. The 'POST' method was utilized for receiving input from HTML input fields and transmitting this data to the server for processing, as illustrated in Figure 26.



```
@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST", "GET"])
```

**Figure 26.** POST and GET methods

The render_template() function was employed as a helper to deliver an HTML template as a response. Functions were defined to facilitate either the return of a value to the HTML page or the rendering of the HTML page itself, as demonstrated in Figure 27 and Table 3.



```
if prediction == 1:
    result = "Slight Injury"
elif prediction == 2:
    result = "Serious Injury"
else:
    result = "Fatal Injury"

return render_template(
    'index.html',
    prediction_text='{} accident chances'.format(result)
)

@flask_app.route("/reset")
def reset():
    return render_template('index.html')

if __name__ == "__main__":
    flask_app.run(debug=True)
```

**Figure 27.** render_template function

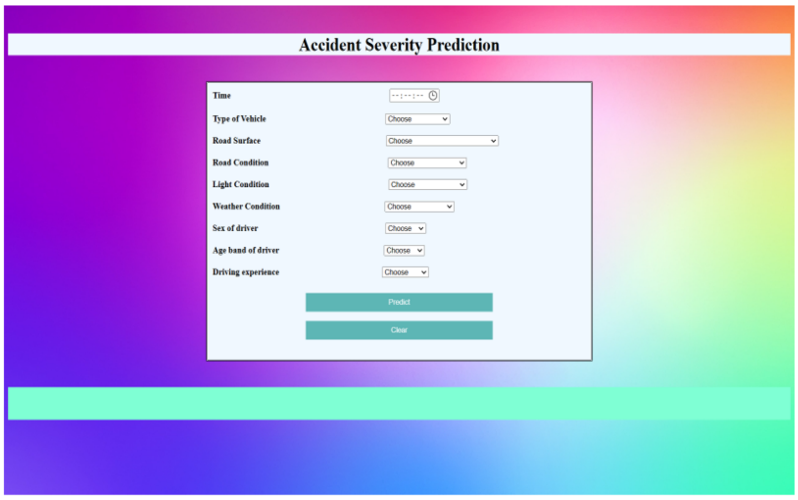
**Table 3.** Attributes

| Name | Details |
|---|---|
| Home() | It is the home page function. |
| Predict() | It is the primary function of getting the I/p from the user from request(), storing it in an array feature by feature, predicting the value, and returning it to the HTML page. |
| Reset() | It redirects to the home page. |

Upon finalizing the coding phase, a comparison of the outputs derived from the various techniques implemented in the coding process was conducted, as depicted in Table 4.

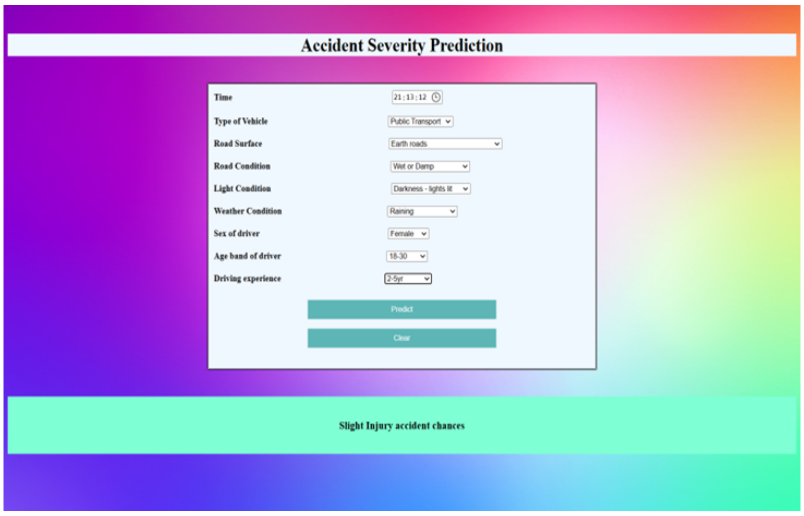
**Table 4.** Attributes

| No. | Technique | Loss Value | Accuracy |
|---|---|---|---|
| 1 | Decision Tree | 0.244 | 83.3% |
| 2 | Random Forest | 0.194 | 77.4% |
| 3 | K-Nearest Neighbour | 0.180 | 82.2% |
| 4 | Naive Bayes | 0.161 | 85.3% |
| 5 | Adaptive Boost Classifier | 0.161 | 85.3% |


Accident Severity Prediction

| | |
|---|---|
| Time | --:--:-- |
| Type of Vehicle | Choose |
| Road Surface | Choose |
| Road Condition | Choose |
| Light Condition | Choose |
| Weather Condition | Choose |
| Sex of driver | Choose |
| Age band of driver | Choose |
| Driving experience | Choose |

**Figure 28.** HTML page and user interface


Accident Severity Prediction

| | |
|---|---|
| Time | 21:13:12 |
| Type of Vehicle | Public Transport |
| Road Surface | Earth roads |
| Road Condition | Wet or Damp |
| Light Condition | Darkness - lights lit |
| Weather Condition | Raining |
| Sex of driver | Female |
| Age band of driver | 18-30 |
| Driving experience | 2-5yr |

Slight Injury accident chances

**Figure 29.** Sample output

Figures 28 and 29 illustrate key aspects of the application under discussion. Figure 28 presents the HTML page and user interface, providing a visual representation of the application's design and layout, which users interact with during operation. Figure 29, on the other hand, shows a sample output from the application. This output provides an example of the results or information that the application produces when the user performs certain actions or inputs certain data.

## 7 Results and Discussion

In the array of algorithms applied to this project, it was observed that AdaBoost and Naive Bayes demonstrated the highest accuracy. Each of these models achieved an accuracy rate of 85.3%, marking them as the most precise among the five models examined. Naive Bayes, a statistical approach, is predominantly utilized in applications such as sentiment analysis, spam filtering, and text classification. Its effectiveness hinges on the assumption of feature independence, excelling particularly when features are independent and the dataset is of high dimensionality. Additionally, its implementation is characterized by relative computational efficiency. However, the performance of Naive Bayes may be compromised if the independence assumption is not met or if interdependencies exist among features.

Conversely, AdaBoost, an ensemble method, constructs a strong classifier by integrating multiple weak classifiers. This method is particularly effective in scenarios involving noisy data where weak classifiers marginally surpass random guesses in classification tasks. Despite its strengths, AdaBoost is sensitive to data noise and outliers and may incur substantial computational costs.

The choice between AdaBoost and Naive Bayes is contingent upon the specific context and data characteristics. Naive Bayes emerges as a suitable option for scenarios with high-dimensional and independent features. On the other hand, AdaBoost is generally more applicable in situations involving noisy data and less effective classifiers.

## 8 Conclusion

The application of ML in predicting the severity of traffic accidents has been established as a significant approach for assessing accident implications. It has been observed that ML models, trained on historical data, possess the capability to categorize accidents into various severity levels, taking into account diverse factors such as weather conditions, vehicle type, time of day, and road surface conditions. An increase in the accuracy of ML models is achievable through the employment of more sophisticated algorithms and the integration of additional variables. The utilization of ML in traffic accident severity prediction offers multiple benefits, including expedited response times, enhanced emergency medical care, and improved overall traffic safety. Accurate predictions are instrumental in formulating policies and strategies aimed at augmenting road safety.

It is imperative to acknowledge that the effectiveness of ML models is intrinsically linked to the quality and reliability of the training data. Ensuring the integrity of this data is crucial for the accuracy of predictions. Moreover, ethical considerations, such as potential biases and privacy concerns associated with using ML in this context, must be meticulously addressed.

In summation, the application of ML in predicting traffic accident severity is a promising strategy that can significantly contribute to traffic safety enhancements and life preservation. The potential of ML in forecasting the severity of future traffic incidents is immense and holds great promise for the field of traffic safety management.

## 9 Future Works

Potential avenues for further research and development include:

(1) Integration of Real-Time Data: Enhancing the precision of predictions can be achieved by training ML models on real-time data encompassing aspects like traffic density, speed, and weather conditions.

(2) Adoption of Advanced Algorithms: The accuracy of forecasts can be improved through the application of advanced ML algorithms, including deep learning and reinforcement learning techniques.

(3) Inclusion of Additional Data Sources: Incorporating diverse data sources, such as social media and smartphone data, can provide a more comprehensive understanding of traffic accidents and their causative factors.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Confilict of Interest

The authors declare that they have no conflicts of interest.

## References

[1] J. Paul, Z. Jahan, K. F. Lateef, M. R. Islam, and S. C. Bakchy, "Prediction of road accident and severity of bangladesh applying machine learning techniques," in *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC)*, Kuching, Malaysia, 2020, pp. 1–6. https://doi.org/10.1109/R10-HTC49770.2020.9356987

[2] I. E. Mallahi, A. Dlia, J. Riffi, M. A. Mahraz, and H. Tairi, "Prediction of traffic accidents using random forest model," in *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2022, pp. 1–7. https://doi.org/10.1109/ISCV54655.2022.9806099

[3] J. Yang, S. Han, and Y. Chen, "Prediction of traffic accident severity based on random forest," *J. Adv. Transp.*, vol. 2023, p. 7641472, 2023. https://doi.org/10.1155/2023/7641472

[4] J. A. Sowdagur, B. T. B. Rozbully-Sowdagur, and G. Suddul, "An artificial neural network approach for road accident severity prediction," in *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia, 2022, pp. 267–270. https://doi.org/10.1109/ZINC55034.2022.9840576

[5] M. F. Labib, A. S. Rifat, M. M. Hossain, A. K. Das, and F. Nawrine, "Road accident analysis and prediction of accident severity by using machine learning in Bangladesh," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, Sarawak, Malaysia, 2019, pp. 1–5. https://doi.org/10.1109/ICSCC.2019.8843640

[6] A. Jadhav, S. Jadhav, A. Jalke, and K. Suryavanshi, "Road accident analysis and prediction of accident severity using machine learning," *Int. Res. J. Eng. Technol.*, vol. 7, pp. 740–747, 2020.

[7] R. Vijithasena and W. Herath, "Data visualization and machine learning approach for analyzing severity of road accidents," in *2022 International Conference for Advancement in Technology (ICONAT)*, Goa, India, 2022, pp. 1–6. https://doi.org/10.1109/ICONAT53423.2022.9726042

[8] J. A. Sowdagur, B. T. B. Rozbully-Sowdagur, and G. Suddul, "Road accident severity prediction in mauritius using supervised machine learning algorithms," in *International Conference on Advances in Computing and Technology (ICACT–2021) Faculty of Computing and Technology (FCT)*, Sri Lanka, 2021, pp. 153–157.

[9] T. T. Bedane, B. G. Assefa, and S. K. Mohapatra, "Preventing traffic accidents through machine learning predictive models," in *2021 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, Bahir Dar, Ethiopia, 2021, pp. 36–41. https://doi.org/10.1109/ICT4DA53266.2021.9672249

[10] Y. Lee, E. Cho, M. Park, H. Kim, K. Choi, and S. Park, "A machine learning approach to prediction of passenger injuries on real road situation," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, Toyama, Japan, 2018, pp. 1087–1090. https://doi.org/10.1109/SCIS-ISIS.2018.00174

[11] S. Koley, S. Mondal, and P. Ghosal, "Smart prediction of severity in vehicular crashes: A machine learning approach," in *2022 5th International Conference on Computational Intelligence and Networks (CINE)*, Bhubaneswar, India, 2022, pp. 1–5. https://doi.org/10.1109/CINE56307.2022.10037474

[12] C. Gutierrez-Osorio and C. Pedraza, "Modern data sources and techniques for analysis and forecast of road accidents: A review," *J. Traffic Transp. Eng.*, vol. 7, pp. 432–446, 2020. https://doi.org/10.1016/j.jtte.2020.05.002

[13] D. Dinesh, "A novel multi-model machine learning approach to real-time road accident prediction and driving behavior analysis," in *2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, Rome, Italy, 2021, pp. 67–72. https://doi.org/10.1109/ISCSIC54682.2021.00023

[14] R. E. Al Mamlook, A. Ali, R. A. Hasan, and H. A. Mohamed Kazim, "Machine learning to predict the freeway traffic accidents-based driving simulation," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, USA, 2019, pp. 630–634. https://doi.org/10.1109/NAECON46414.2019.9058268

[15] D. Al-Dogom, N. Aburaed, M. Al-Saad, and S. Almansoori, "Spatio-temporal analysis and machine learning for traffic accidents prediction," in *2019 2nd International Conference on Signal Processing and Information Security (ICSPIS)*, Dubai, United Arab Emirates, 2019, pp. 1–4. https://doi.org/10.1109/ICSPIS48135.2019.9045892

[16] E. Suganya and S. Vijayarani, "Analysis of road accidents in India using data mining classification algorithms," in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, Coimbatore, India, 2017, pp. 1122–1126. https://doi.org/10.1109/ICICI.2017.8365315

[17] M. Manzoor, M. Umer, S. Sadiq, A. Ishaq, S. Ullah, H. A. Madni, and C. Bisogni, "RFCNN: Traffic accident severity prediction based on decision level fusion of machine and deep learning model," *IEEE Access*, vol. 9, pp. 128 359–128 371, 2021. https://doi.org/10.1109/ACCESS.2021.3112546

[18] P. Tiwari, S. Kumar, and D. Kalitin, "Road-user specific analysis of traffic accident using data mining techniques," in *Computational Intelligence, Communications, and Business Analytics (CICBA): First International Conference*, Kolkata, India, 2017, pp. 398–410. https://doi.org/10.1007/978-981-10-6430-2_31