CCS1304 - Business Intelligence

Group Assignment 01

Predicting Accident Severity Based on Driver Behaviors and Environmental Factors

22ug2-0119	Sanduni Chethana	22ug2-0119@sltc.ac.lk
22ug2-0023	Thavisha Nipun	22ug2-0023@sltc.ac.lk
22ug2-0017	Deshani Wijewardhana	22ug2-0017@sltc.ac.lk
22ug2-0271	Muthuni Nimshi	22ug2-0271@sltc.ac.lk
22ug2-0068	Nethulya Sooriarachchi	22ug2-0068@sltc.ac.lk

Table of Contents

1. Problem Statement

Traffic accidents continue to be a major threat to public safety with loss of lives, injuries, and disruption to the economy. Knowledge of those underlying factors that may affect the severity of accidents will graciously play a role in granting authorities the ability to take proactive and effective safety measures.

The present study attempts to study the application of machine learning with an emphasis on predicting severity in terms of Low, Moderate, or High according to driver behavior and Environmental factors. The end result should be a data-driven model that will go a long way in achieving considerable road safety and assist traffic management systems to incorporate such information in their decision-making processes.

2. Problem Description

Traffic accidents can generally get more or less severe with driver behaviors such as Alcohol level of the driver , Driver experience and age of the driver sand environmental factors, such as weather conditions , road conditions , road type, road light conditions and traffic density. Conventional safety measures have relied on historical accident reports and expert judgments, while machine learning presents an approach that is more accurate and scalable.

Key research questions are as follows.

- How do environmental factors impact accident severity?
- How do driver behaviours impact accident severity?
- Moreover, is it even possible, given the historic data, to train a machine-learning model that gives some sensible predictions of accident severity?
- Which machine learning model serves with maximum accuracy in the classification of severity?

This study applies some different machine-learning algorithms to predict the severity of an accident, assesses the algorithms' performance, and tries to arrive at an optimal model for predicting accident severity.

3. About the Data Set

This dataset contains data designed to predict the severity of traffic accidents based on various factors like road conditions, driver behavior, and traffic situations.In this data set filled with both categorical and numerical data.

Categorical Data

Weather: The impact of weather conditions on the likelihood of accidents.

- Clear: No adverse weather conditions.
- Rainy: Rainy conditions increase the chance of accidents.
- Foggy: Foggy conditions reduce visibility, increasing accident chances.
- Snowy: Snow can cause slippery roads and higher accident probability.
- Stormy: Stormy weather can create hazardous driving conditions.

Road_Type: The type of road, influencing the probability of accidents.

- Highway: High-speed roads with higher chances of severe accidents.
- City Road: Roads within city limits, typically with more traffic and lower speeds.
- Rural Road: Roads outside urban areas, often with fewer vehicles and lower speeds.
- Mountain Road: Roads with curves and elevation changes, increasing accident risk.

Time_of_Day: The time of day when the accident occurs.

- Morning: The period between sunrise and noon.
- Afternoon: The period between noon and evening.
- Evening: The period just before sunset.
- Night: The nighttime, often associated with reduced visibility and higher risk.

Accident_Severity: The severity of the accident.

- Low: Minor accident.
- Moderate: Moderate accident with some damage or injuries.
- High: Severe accident with significant damage or injuries.

Road_Condition: The condition of the road surface.

- Dry: Dry roads with minimal risk.
- Wet: Wet roads due to rain, increasing the risk of accidents.
- Icy: Ice on the road, significantly increasing the risk of accidents.
- Under Construction: Roads under construction, which may have obstacles or poor road quality.

Vehicle_Type: The type of vehicle involved in the accident.

- Car: A regular passenger car.
- Truck: A large vehicle used for transporting goods.
- Motorcycle: A two-wheeled motor vehicle.
- Bus: A large vehicle used for public transportation.

Road_Light_Condition: The lighting conditions on the road.

- Daylight: Daytime, when visibility is typically good.
- Artificial Light: Road is illuminated with streetlights.
- No Light: Road is not illuminated, typically during the night in poorly lit areas.

Numerical Data

Traffic_Density: The level of traffic on the road.

- 0: Low density (few vehicles).
- 1: Moderate density.
- 2: High density (many vehicles).

Speed_Limit: The maximum allowed speed on the road.

Number_of_Vehicles: The number of vehicles involved in the accident, ranging from 1 to 5.

Driver_Alcohol: Whether the driver consumed alcohol.

- 0: No alcohol consumption.
- 1: Alcohol consumption (which increases the likelihood of an accident).

Driver_Age: The age of the driver. Values range from 18 to 70 years old.

Driver_Experience: The years of experience the driver has. Values range from 0 to 50 years of experience.

4. Methodology

This project follows a structured data science workflow including methods like Data preprocessing , Data analysing / Data visualization, Model selection and training and Model evaluation to develop an effective predictive model.

4.1 Data Preprocessing

- Handling Missing Values

  Check for missing data and apply imputation techniques where necessary. Filling numerical values with median and filling categorical values with mode.

```
#Filling Numerical values

df['Traffic_Density'].fillna(df['Traffic_Density'].median(), inplace=True)
df['Speed_Limit'].fillna(df['Speed_Limit'].median(), inplace=True)
df['Number_of_Vehicles'].fillna(df['Number_of_Vehicles'].median(), inplace=True)
df['Driver_Alcohol'].fillna(df['Driver_Alcohol'].median(), inplace=True)
df['Driver_Age'].fillna(df['Driver_Age'].median(), inplace=True)
df['Driver_Experience'].fillna(df['Driver_Experience'].median(), inplace=True)
df['Accident'].fillna(df['Accident'].median(), inplace=True)
```

```
#Filling categorical data

df['Weather'] = df['Weather'].fillna(df['Weather'].mode()[0])
df['Road_Type'] = df['Road_Type'].fillna(df['Road_Type'].mode()[0])
df['Time_of_Day'] = df['Time_of_Day'].fillna(df['Time_of_Day'].mode()[0])
df['Accident_Severity'] = df['Accident_Severity'].fillna(df['Accident_Severity'].mode()[0])
df['Road_Condition'] = df['Road_Condition'].fillna(df['Road_Condition'].mode()[0])
df['Vehicle_Type'] = df['Vehicle_Type'].fillna(df['Vehicle_Type'].mode()[0])
df['Road_Light_Condition'] = df['Road_Light_Condition'].fillna(df['Road_Light_Condition'].mode()[0])
```

- Handling Duplicates

  Count all duplicates and drop them.

```
[ ]  duplicates_count = df.duplicated().sum()
     print(f"Duplicates count: {duplicates_count}")

     Duplicates count: 18

[ ]  df.drop_duplicates(inplace=True)

[ ]  duplicates_count = df.duplicated().sum()
     print(f"Duplicates count: {duplicates_count}")

     Duplicates count: 0
```

- Encoding Categorical Variables

  Convert categorical variables into numerical representations using one-hot encoding .

```
from sklearn.preprocessing import OneHotEncoder

# One-hot encoding for categorical features
X_environmental = pd.get_dummies(X_environmental, drop_first=True)
X_driver = pd.get_dummies(X_driver, drop_first=True)
```

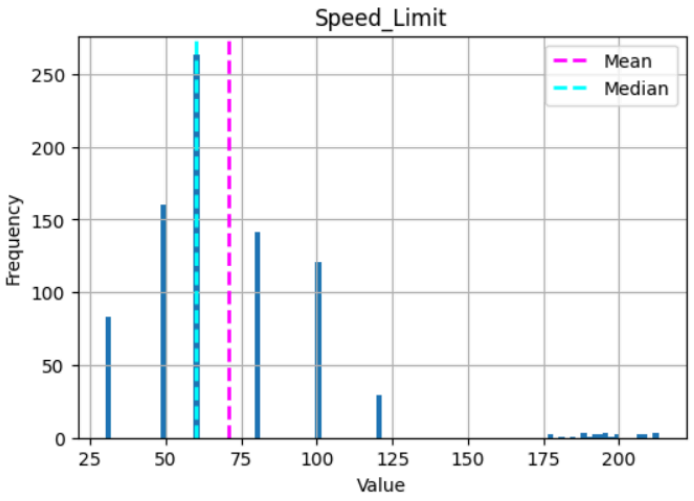
4.1 Data Visualization

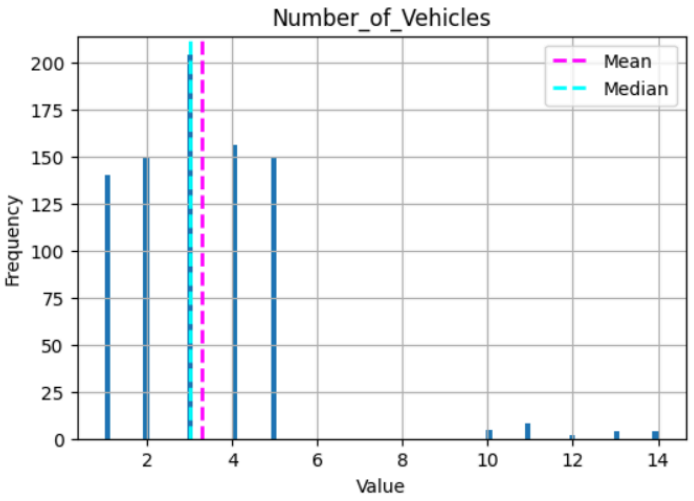
- Visualize Feature Distributions

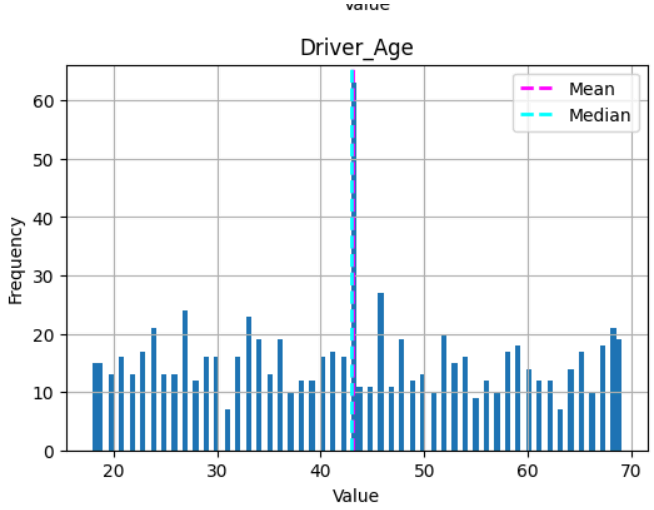
  Histograms for all numerical & categorical features to identify the distribution of the data.Each of these histograms contains mean and the median of the each feature

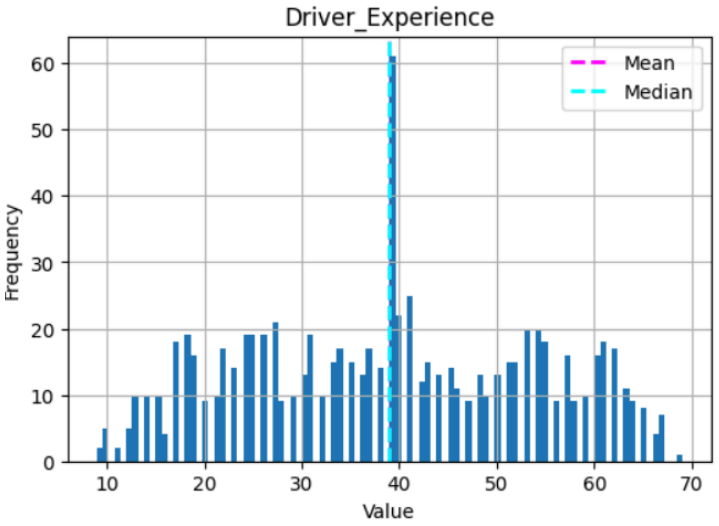
  Numerical Variables

```
# Numerical columns
Numerical_col = ['Speed_Limit', 'Number_of_Vehicles','Driver_Age', 'Driver_Experience', 'Accident']

# Plot a histogram for each numeric column
for col in Numerical_col:
    fig = plt.figure(figsize=(6, 4))
    ax = fig.gca()
    feature = df[col]

    # Plot histogram
    feature.hist(bins=100, ax=ax)

    # Add vertical lines for mean and median
    ax.axvline(feature.mean(), color='magenta', linestyle='dashed', linewidth=2, label="Mean")
    ax.axvline(feature.median(), color='cyan', linestyle='dashed', linewidth=2, label="Median")

    # Titles and labels
    ax.set_title(col)
    ax.set_xlabel("Value")
    ax.set_ylabel("Frequency")  # Set y-axis label
    ax.legend()

plt.show()
```



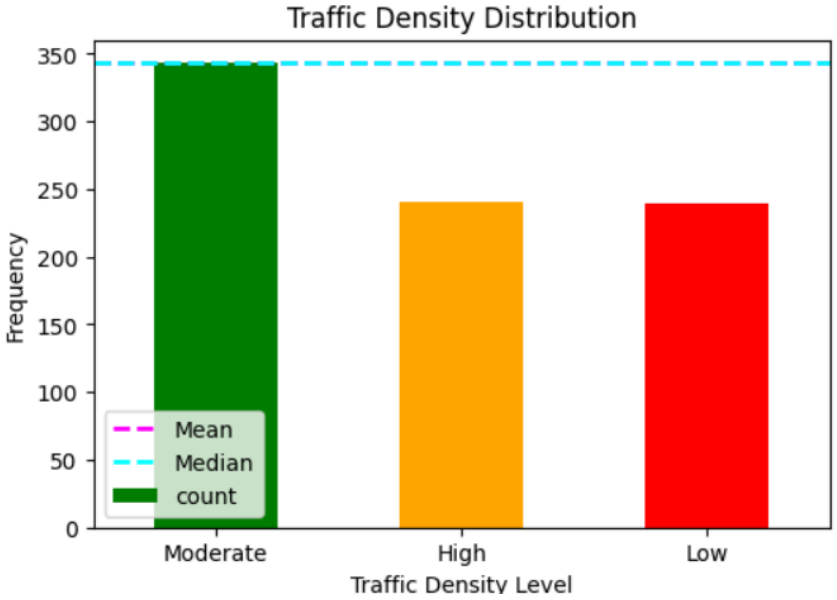






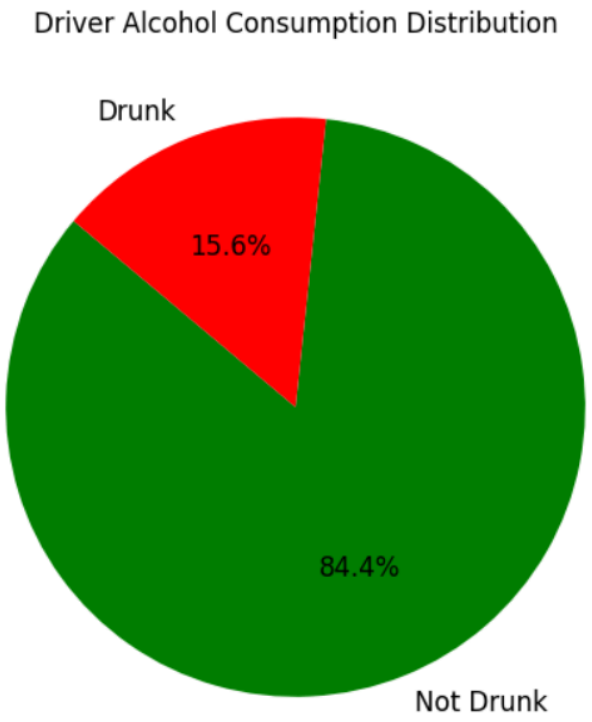
Plot for Traffic Density distribution

```
# Define category labels
traffic_labels = {0: 'Low', 1: 'Moderate', 2: 'High'}

# Map numeric values to categorical labels
df['Traffic_Density_Category'] = df['Traffic_Density'].map(traffic_labels)

# Count occurrences of each category
density_counts = df['Traffic_Density_Category'].value_counts()

# Plot bar chart
fig, ax = plt.subplots(figsize=(6, 4))
bars = density_counts.plot(kind='bar', color=['green', 'orange', 'red'], ax=ax)

# Calculate mean and median as numerical values
mean_value = df['Traffic_Density'].mean()
median_value = df['Traffic_Density'].median()

# Convert numerical mean/median to category positions
mean_category = traffic_labels[int(round(mean_value))]
median_category = traffic_labels[int(round(median_value))]

# Add mean and median lines
ax.axhline(y=density_counts[mean_category], color='magenta', linestyle='dashed', linewidth=2, label='Mean')
ax.axhline(y=density_counts[median_category], color='cyan', linestyle='dashed', linewidth=2, label='Median')

# Labels and title
ax.set_title("Traffic Density Distribution")
ax.set_xlabel("Traffic Density Level")
ax.set_ylabel("Frequency")
plt.xticks(rotation=0)
ax.legend()
```



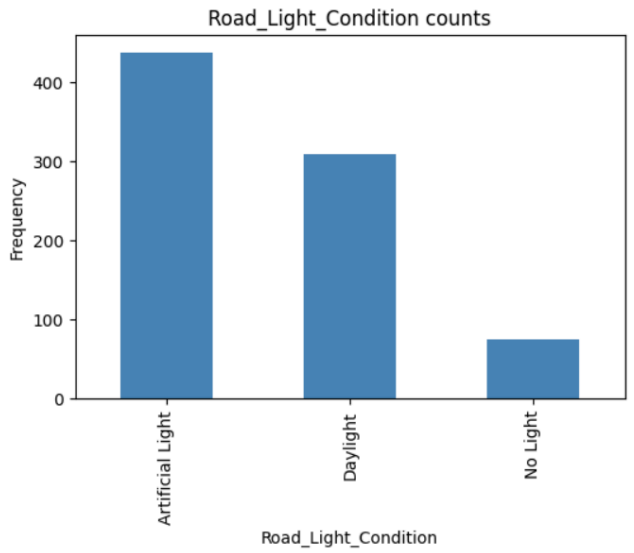
Pie chart for driver alcohol consumption distribution

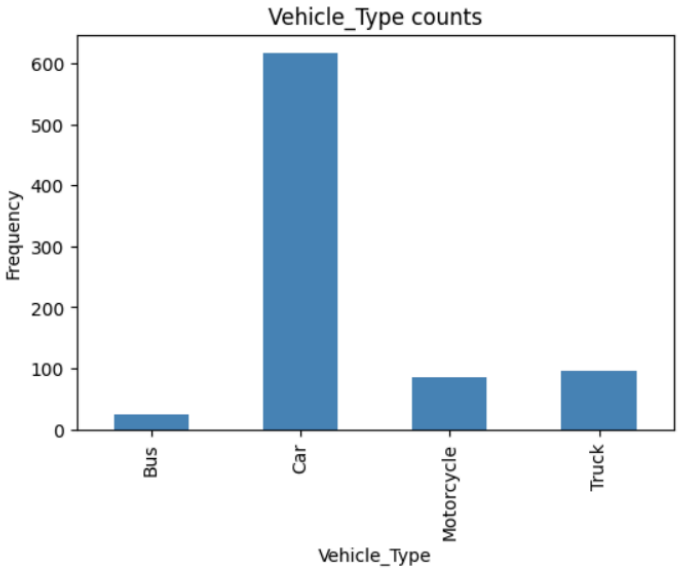
```
import matplotlib.pyplot as plt

# Define category labels
alcohol_labels = {0: 'Not Drunk', 1: 'Drunk'}
# Map numeric values to categorical labels
df['Driver_Alcohol_Category'] = df['Driver_Alcohol'].map(alcohol_labels)

# Count occurrences of each category
alcohol_counts = df['Driver_Alcohol_Category'].value_counts()

# Define colors: Green for "Not Drunk", Red for "Drunk"
colors = {'Not Drunk': 'green', 'Drunk': 'red'}

# Plot pie chart
fig, ax = plt.subplots(figsize=(6, 6))
wedges, texts, autotexts = ax.pie(
    alcohol_counts, labels=alcohol_counts.index, autopct='%1.1f%%',
    colors=[colors[label] for label in alcohol_counts.index],
    startangle=140, textprops={'fontsize': 12}
)

# Title
ax.set_title("Driver Alcohol Consumption Distribution")
plt.show()
```

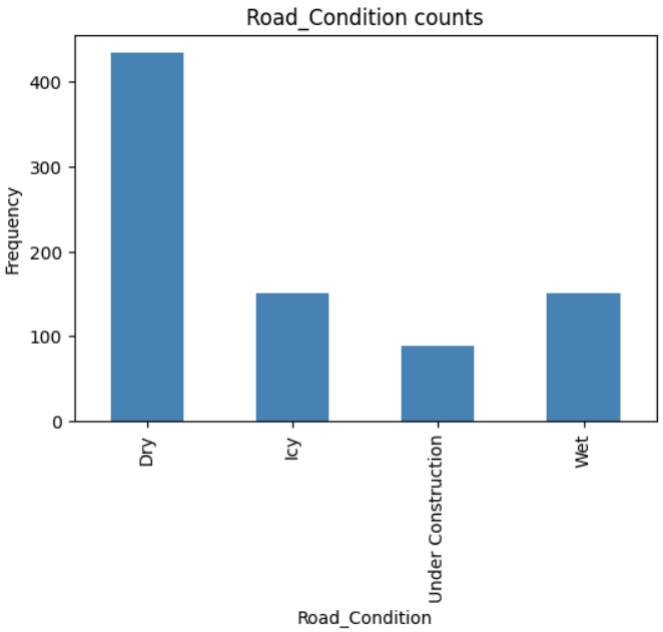


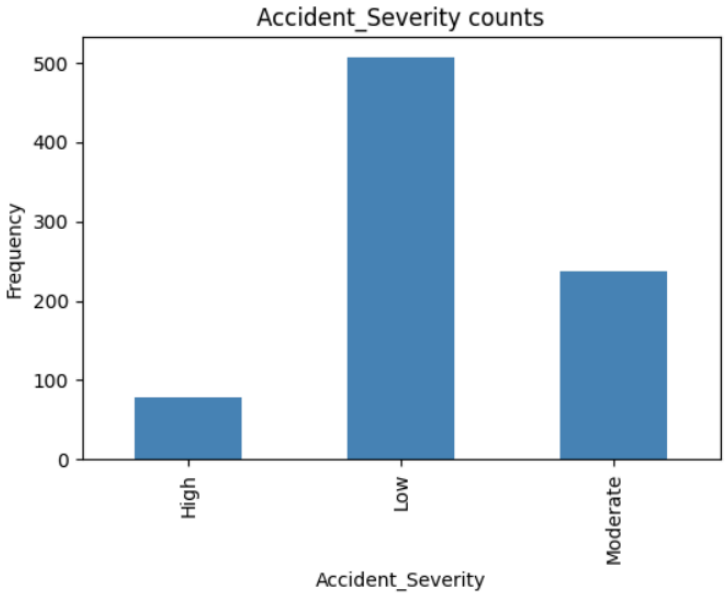
Plots for each categorical Feature

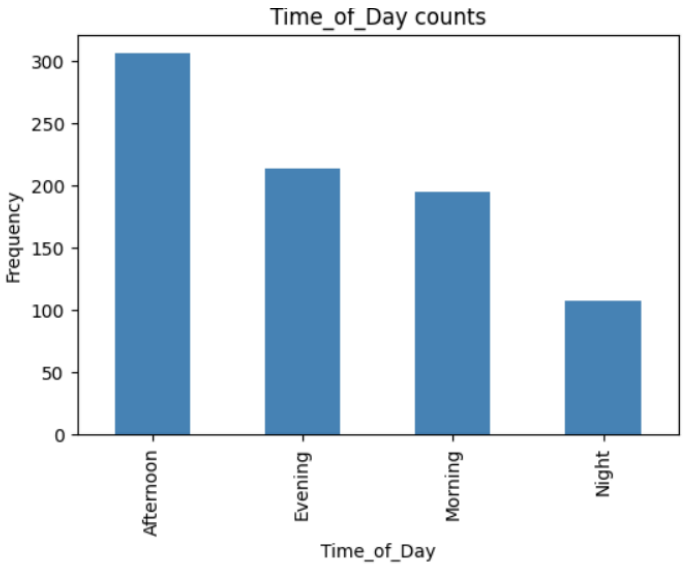
```
# Plot a bar plot for each categorical feature count
categorical_features = ['Weather', 'Road_Type', 'Time_of_Day', 'Accident_Severity', 'Road_Condition', 'Vehicle_Type', 'Road_Light_Condition']

for col in categorical_features:
    counts = df[col].value_counts().sort_index()
    fig = plt.figure(figsize=(6, 4))
    ax = fig.gca()
    counts.plot.bar(ax = ax, color='steelblue')
    ax.set_title(col + ' counts')
    ax.set_xlabel(col)
    ax.set_ylabel("Frequency")
plt.show()
```

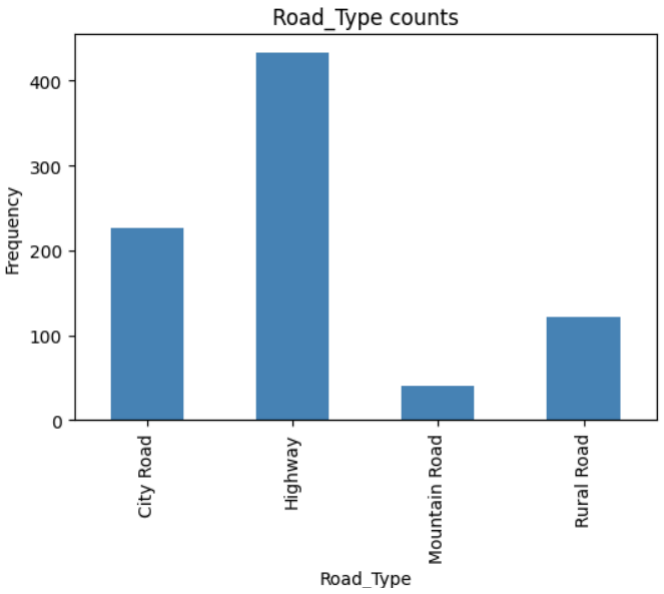


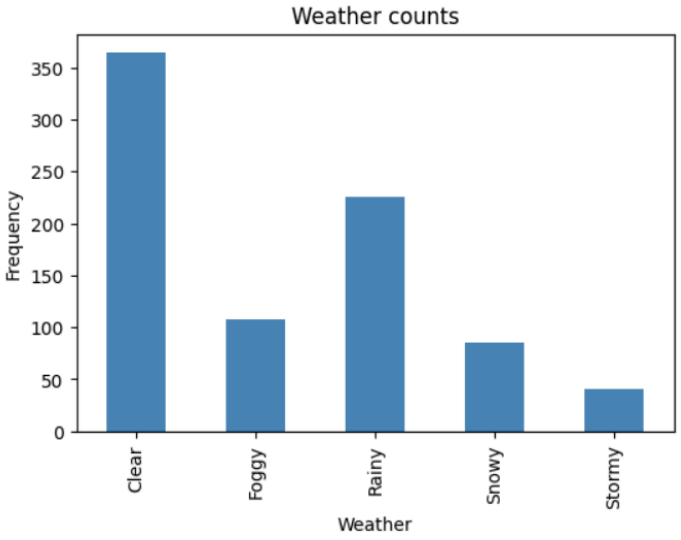












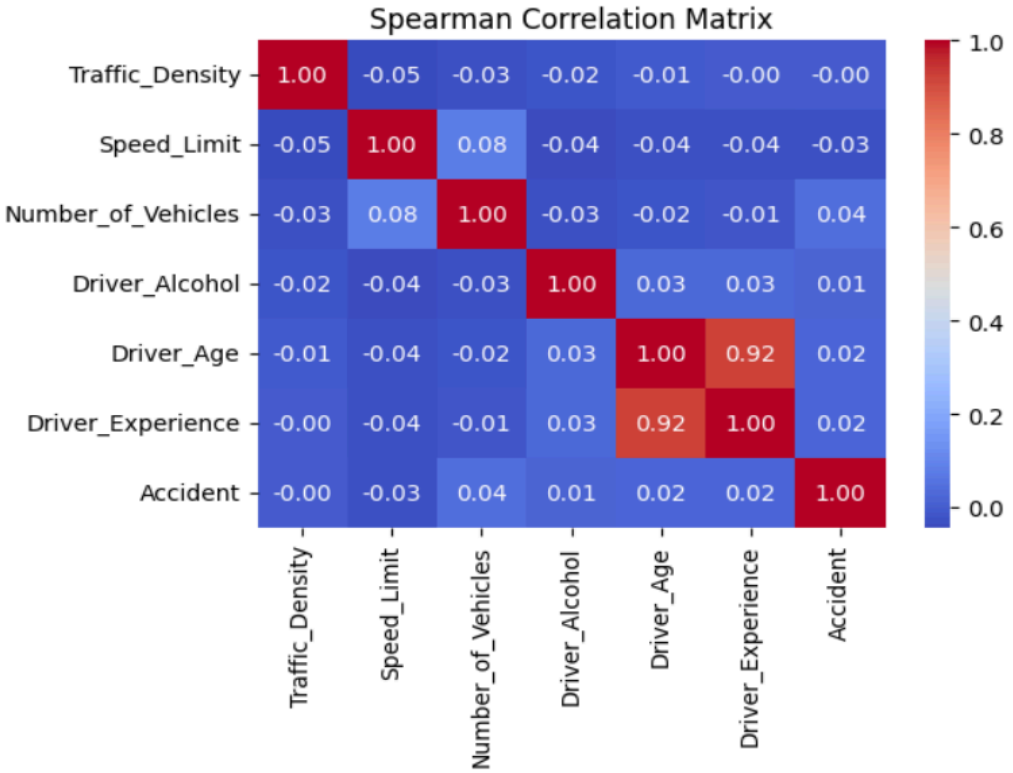
- Analyze Correlations:

  Heatmap and correlation matrices to identify key factors influencing accident severity.

```
numeric_df = df.select_dtypes(include=['number'])
spearman_corr = numeric_df.corr(method='spearman')

# printing corralation of numerical values
print(spearman_corr)

#Heat map for Numerical values
plt.figure(figsize=(8, 7))
sns.heatmap(spearman_corr, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title('Spearman Correlation Matrix')
plt.show()
```

	Traffic_Density	Speed_Limit	Number_of_Vehicles	\
Traffic_Density	1.000000	-0.045265	-0.030647	
Speed_Limit	-0.045265	1.000000	0.076297	
Number_of_Vehicles	-0.030647	0.076297	1.000000	
Driver_Alcohol	-0.022652	-0.037757	-0.030421	
Driver_Age	-0.005930	-0.036819	-0.024255	
Driver_Experience	-0.002173	-0.035221	-0.013461	
Accident	-0.001008	-0.028651	0.036608	

	Driver_Alcohol	Driver_Age	Driver_Experience	Accident
Traffic_Density	-0.022652	-0.005930	-0.002173	-0.001008
Speed_Limit	-0.037757	-0.036819	-0.035221	-0.028651
Number_of_Vehicles	-0.030421	-0.024255	-0.013461	0.036608
Driver_Alcohol	1.000000	0.025207	0.026769	0.012788
Driver_Age	0.025207	1.000000	0.924057	0.022875
Driver_Experience	0.026769	0.924057	1.000000	0.022151
Accident	0.012788	0.022875	0.022151	1.000000



4.3 Model Selection and Training

First separate features and as environmental and driver features then train and split separated data. After that machine learning models were used to predict accident severity

Separate Features

```
environmental_features = ['Weather', 'Road_Type', 'Time_of_Day',
                          'Road_Condition', 'Traffic_Density' ,'Road_Light_Condition']
X_environmental = df[environmental_features]
y = df['Accident_Severity']

driver_features = ['Vehicle_Type','Driver_Alcohol', 'Driver_Age', 'Driver_Experience']
X_driver = df[driver_features]
y = df['Accident_Severity']
```

Train Test and Split Data

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# One-hot encoding for categorical features
X_environmental = pd.get_dummies(X_environmental, drop_first=True)
X_driver = pd.get_dummies(X_driver, drop_first=True)

# Split data into train and test sets (80% train, 20% test)
X_env_train, X_env_test, y_train, y_test = train_test_split(X_environmental, y, test_size=0.2, random_state=42)
X_driver_train, X_driver_test, y_train, y_test = train_test_split(X_driver, y, test_size=0.2, random_state=42)
```

- ML model 1 - Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Logistic Regression for environmental features
log_reg_env = LogisticRegression(max_iter=1000)
log_reg_env.fit(X_env_train, y_train)
y_pred_env_log_reg = log_reg_env.predict(X_env_test)
acc_env_log_reg = accuracy_score(y_test, y_pred_env_log_reg)

# Logistic Regression for driver features
log_reg_driver = LogisticRegression(max_iter=1000)
log_reg_driver.fit(X_driver_train, y_train)
y_pred_driver_log_reg = log_reg_driver.predict(X_driver_test)
acc_driver_log_reg = accuracy_score(y_test, y_pred_driver_log_reg)

# Print Accuracy
print("Logistic Regression Accuracy (Environmental):", acc_env_log_reg)
print("Logistic Regression Accuracy (Driver):", acc_driver_log_reg)

# Print Classification Reports
print("\nClassification Report (Environmental Factors):\n", classification_report(y_test, y_pred_env_log_reg))
print("\nClassification Report (Driver Behavior):\n", classification_report(y_test, y_pred_driver_log_reg))
```

- ML model 2 - Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Random Forest for environmental features
rf_env = RandomForestClassifier(random_state=42)
rf_env.fit(X_env_train, y_train)
y_pred_env_rf = rf_env.predict(X_env_test)
print("Random Forest Accuracy (Environmental):", accuracy_score(y_test, y_pred_env_rf))

# Random Forest for driver features
rf_driver = RandomForestClassifier(random_state=42)
rf_driver.fit(X_driver_train, y_train)
y_pred_driver_rf = rf_driver.predict(X_driver_test)
print("Random Forest Accuracy (Driver):", accuracy_score(y_test, y_pred_driver_rf))

print("\nClassification Report (Environmental Factors):\n", classification_report(y_test, y_pred_env_rf))
print("\nClassification Report (Driver Behavior):\n", classification_report(y_test, y_pred_driver_rf))
```

- ML model 3 - Decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier

# Decision Tree for environmental features
dt_env = DecisionTreeClassifier(random_state=42)
dt_env.fit(X_env_train, y_train)
y_pred_env_dt = dt_env.predict(X_env_test)
print("Decision Tree Accuracy (Environmental):", accuracy_score(y_test, y_pred_env_dt))

# Decision Tree for driver features
dt_driver = DecisionTreeClassifier(random_state=42)
dt_driver.fit(X_driver_train, y_train)
y_pred_driver_dt = dt_driver.predict(X_driver_test)
print("Decision Tree Accuracy (Driver):", accuracy_score(y_test, y_pred_driver_dt))

# Print Classification Reports
print("\nClassification Report (Environmental Factors):\n", classification_report(y_test, y_pred_env_dt))
print("\nClassification Report (Driver Behavior):\n", classification_report(y_test, y_pred_driver_dt))
```

4.4 Model Evaluation

- Performance metrics like Accuracy, Precision, Recall, and F1-score to evaluate Each model and to select the most accurate one.

```
from sklearn.metrics import accuracy_score, classification_report
```

Note - Information on performances are included in the results section.

5 . Results

Accuracy and Classification reports of each model

1. ML model 1 - Logistic Regression

```
Logistic Regression Accuracy (Environmental): 0.6606060606060606
Logistic Regression Accuracy (Driver): 0.6787878787878788

Classification Report (Environmental Factors):
              precision    recall  f1-score   support

        High       0.00      0.00      0.00        10
         Low       0.68      0.97      0.80       112
    Moderate       0.00      0.00      0.00        43

    accuracy                           0.66       165
   macro avg       0.23      0.32      0.27       165
weighted avg       0.46      0.66      0.54       165


Classification Report (Driver Behavior):
              precision    recall  f1-score   support

        High       0.00      0.00      0.00        10
         Low       0.68      1.00      0.81       112
    Moderate       0.00      0.00      0.00        43

    accuracy                           0.68       165
   macro avg       0.23      0.33      0.27       165
weighted avg       0.46      0.68      0.55       165
```

2. ML model 2 - Random Forest Classifier

```
Random Forest Accuracy (Environmental): 0.6121212121212121
Random Forest Accuracy (Driver): 0.503030303030303

Classification Report (Environmental Factors):
              precision    recall  f1-score   support

        High       0.00      0.00      0.00        10
         Low       0.69      0.81      0.75       112
    Moderate       0.40      0.23      0.29        43

    accuracy                           0.61       165
   macro avg       0.36      0.35      0.35       165
weighted avg       0.58      0.61      0.59       165


Classification Report (Driver Behavior):
              precision    recall  f1-score   support

        High       0.00      0.00      0.00        10
         Low       0.65      0.65      0.65       112
    Moderate       0.24      0.23      0.24        43

    accuracy                           0.50       165
   macro avg       0.30      0.29      0.30       165
weighted avg       0.51      0.50      0.50       165
```

3. ML model 3 - Decision tree classifier classification

```
Decision Tree Accuracy (Environmental): 0.509090909090909
Decision Tree Accuracy (Driver): 0.4909090909090909

Classification Report (Environmental Factors):
              precision    recall  f1-score   support

        High       0.05      0.10      0.07        10
         Low       0.67      0.64      0.65       112
    Moderate       0.30      0.26      0.28        43

    accuracy                           0.51       165
   macro avg       0.34      0.33      0.33       165
weighted avg       0.53      0.51      0.52       165


Classification Report (Driver Behavior):
              precision    recall  f1-score   support

        High       0.00      0.00      0.00        10
         Low       0.68      0.61      0.64       112
    Moderate       0.30      0.30      0.30        43

    accuracy                           0.49       165
   macro avg       0.33      0.30      0.31       165
weighted avg       0.54      0.49      0.51       165
```

- Within all above ML Models highest accuracy contains Logistic Regression Model . It provides accuracy of 66.1% when predicting accident severity based on environmental factors and provides accuracy of 67.78% when predicting accident severity based on Driver behaviours.
- The findings showed that driver behaviors such as Alcohol level of the driver , Driver experience and age of the driver , and environmental factors, such as weather conditions , road conditions , road type, road light conditions and traffic density, were significant predictors of severity.

6 . Discussion

6.1 Limitations

- Limited Feature Scope - The dataset does not include external factors such as driver fatigue, distractions like whether the driver is using the mobile or not , or road maintenance history.
- Class Imbalance - If severe accidents are rare, models may struggle to predict them accurately. Techniques like oversampling or adjusting class weights may help.
- Data Bias - The dataset may not fully represent real-world accident patterns in different regions.

6.2 Future Improvements

- Expand Data - Collect more real-world accident data, including additional risk factors.
- Advanced Models: Explore deep learning approaches such as Recurrent Neural Networks (RNNs) or Transformer-based models for sequential pattern recognition.
- Real-Time Prediction: Implement real-time accident severity prediction for emergency response systems.
- Geospatial Analysis: Incorporate geographical data to identify accident-prone zones and optimize traffic management.

7. Conclusion

This study successfully developed a machine learning model to predict accident severity using environmental, traffic, and driver-related factors. The findings showed that driver behaviours such as the Alcohol level of the driver, Driver experience and age of the driver, and environmental factors, such as weather conditions, road conditions, road type, road light conditions and traffic density, were significant predictors of severity. The best-performing model is Logistic Regression can be used for predictive road safety applications. Future work will focus on improving data collection, incorporating real-time analysis, and integrating the model into intelligent transportation systems.