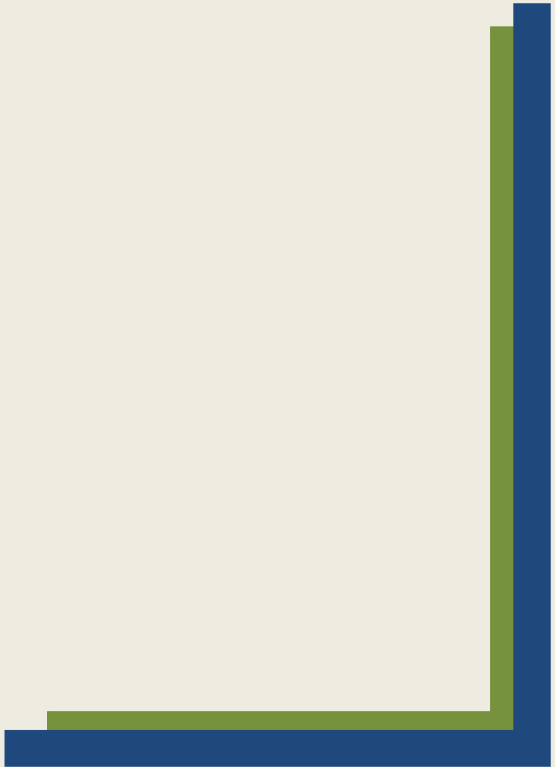




RANDOM FOREST

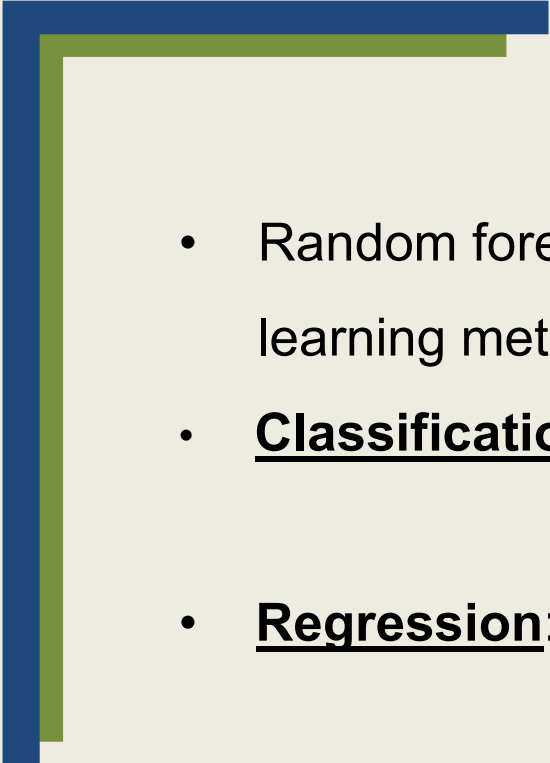
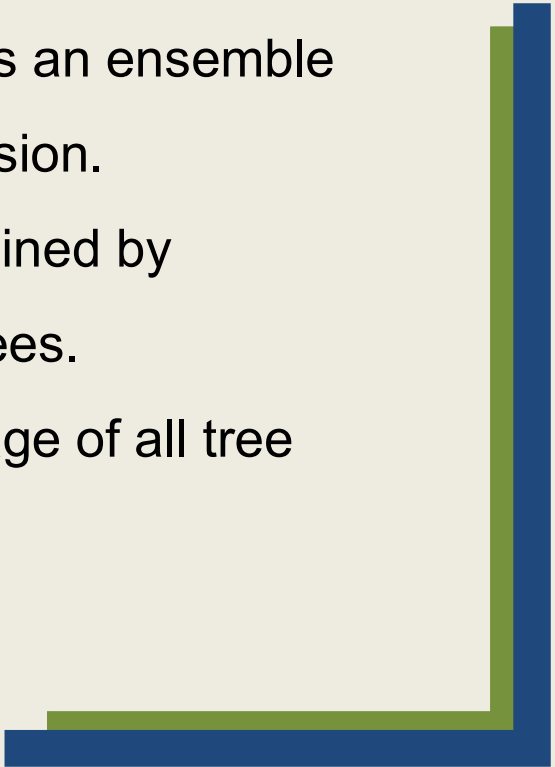


GROUP 1:

1. 23CE10032 Modalavalasa Keerthi
 2. 23CE10033 Mudam Sneha
 3. 23CE10034 Nalla Shivatej Reddy
- 

What is Random Forest ?

- Random Forest is an ensemble method that constructs a "forest" of decision trees during training.
- Each tree in the forest is built using a random subset of the data and features, and the final prediction is obtained by aggregating the predictions of all the trees.

- 
- 
- Random forests or random decision forests is an ensemble learning method for classification and regression.
 - **Classification**: The final prediction is determined by majority voting among the trees.
 - **Regression**: The final prediction is the average of all tree predictions.

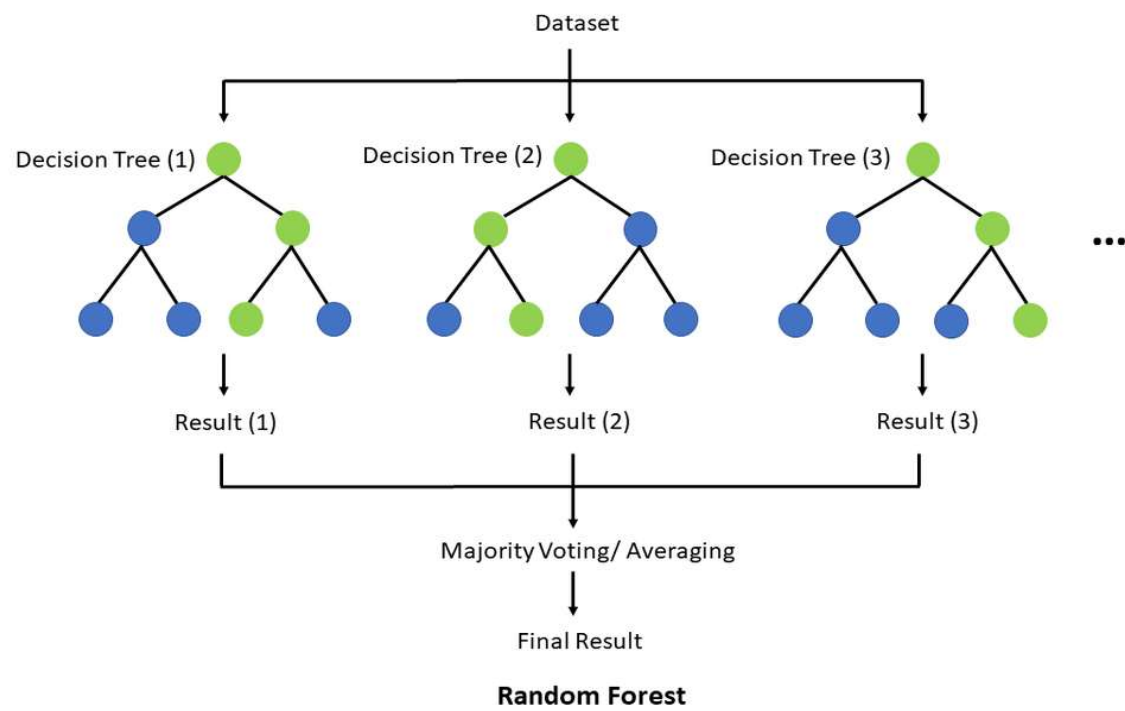
How Random Forest Works ?

Training Phase:

- Generate multiple bootstrap samples from the original training dataset .
- Build a decision tree for each bootstrap sample.
- At each node of the tree, use a random subset of features to determine the best split.

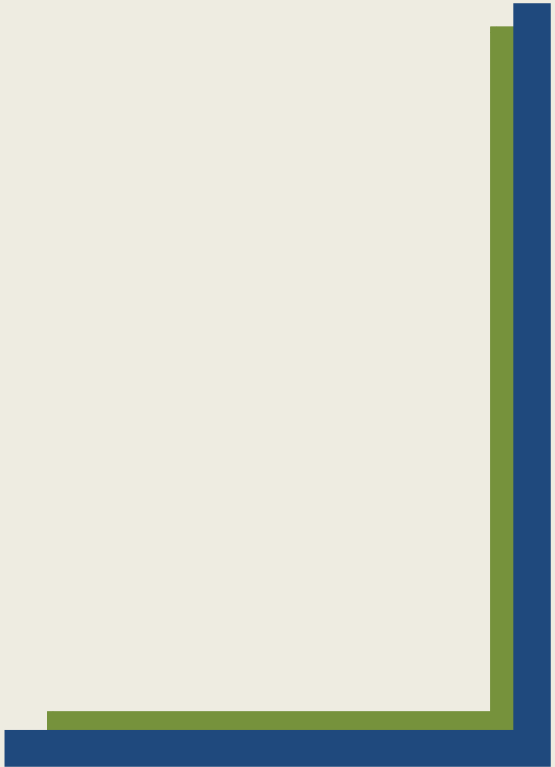
Prediction Phase:

- For classification, each tree votes for a class, and the majority class is selected.
- For regression, the predictions from all trees are averaged.





Applications in Transportation :

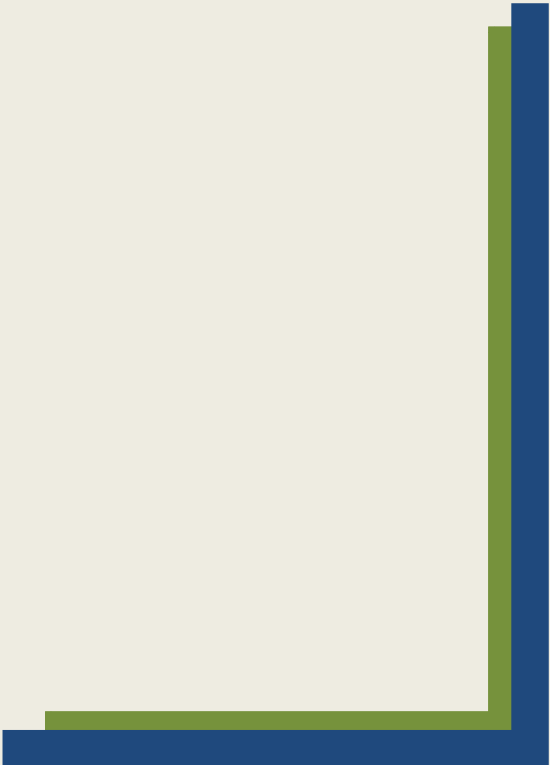
1. Traffic Accident Analysis
 2. Traffic Flow Prediction
 3. Road Safety Management
 4. Public Transportation
 5. Infrastructure Planning
- 



Advantages:

- High accuracy
- Works with mixed data types
- Handles overfitting well
- Provides feature importance

Disadvantage:

- Computationally expensive
 - Slower predictions for large models
 - May struggle with high-dimensional data
- 

Problem Statement

"Using the Random Forest algorithm, we aim to find the Occurrence and Accident Severity based on the following factors:

- **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.

- **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.
- **Driver_Alcohol:** Whether the driver consumed alcohol.

0: No alcohol consumption.

1: Alcohol consumption (which increases the likelihood of an accident).

- **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.

- **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.
- **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.

- **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.

- The outputs are Confusion Matrix of “ Accident Occurrence” Prediction and “Accident Severity” Prediction.
- Using The Confusion Matrix And also it gives the Accuracy of Model In finding “ Accident Occurrence” and “Accident Severity”.
- Structure of a Confusion Matrix

Actual \ Predicted	Predicted No (0)	Predicted Yes (1)
Actual No (0)	True Negative (TN)	False Positive (FP)
Actual Yes (1)	False Negative (FN)	True Positive (TP)

- Finding Accuracy Using Confusion Matrix

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

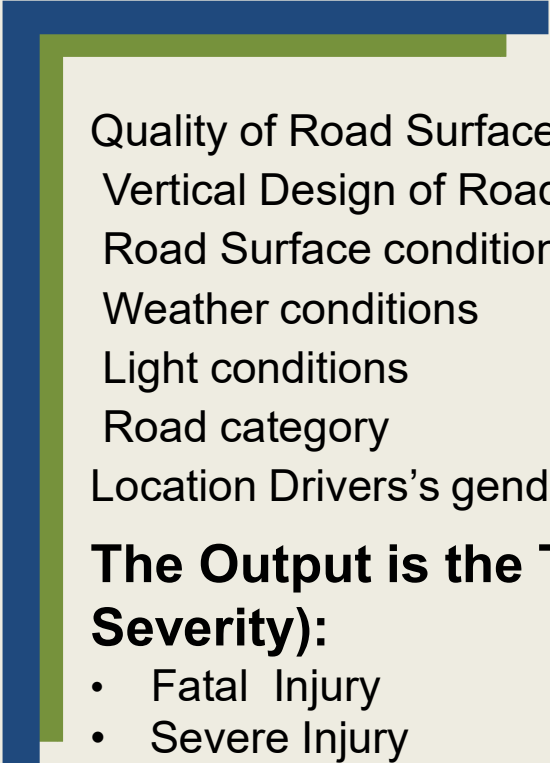
Article :

“Classification of Driver Injury Severity for Accidents Involving Heavy Vechiles with Decision Tree and Random Forest”

- This study implies that the variables associated with categories of injury severity can be referred by road safety practitioners to plan for the best measures needed in reducing road fatalities, especially among heavy vehicle drivers.

The Input Variables :

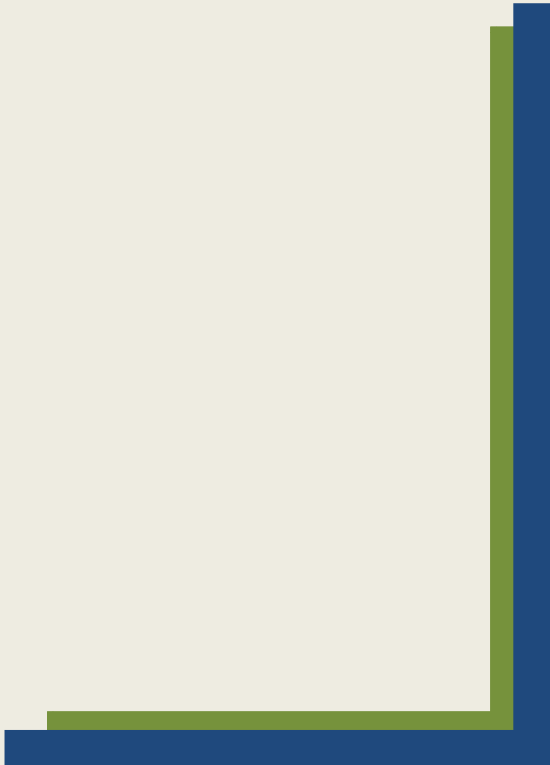
- Number of vehicles involved
- Type of first collision
- Road surface



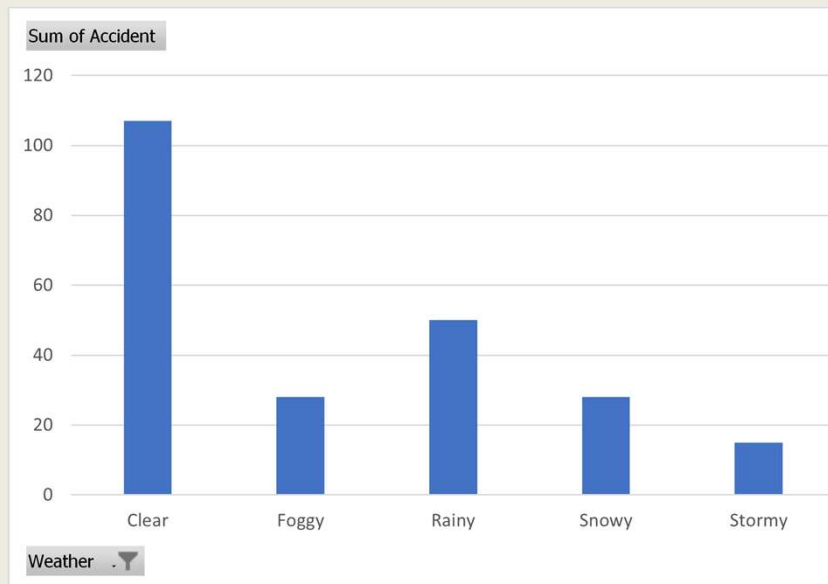
Quality of Road Surface
Vertical Design of Road
Road Surface condition
Weather conditions
Light conditions
Road category
Location Drivers's gender

Surrounding area
Heavy vechile type
Vechile modification
Driver's age
Driver's error
Drunk driving

The Output is the Type Of Injury(Level of Severity):

- Fatal Injury
 - Severe Injury
 - Slight Injury
 - No injury
- 

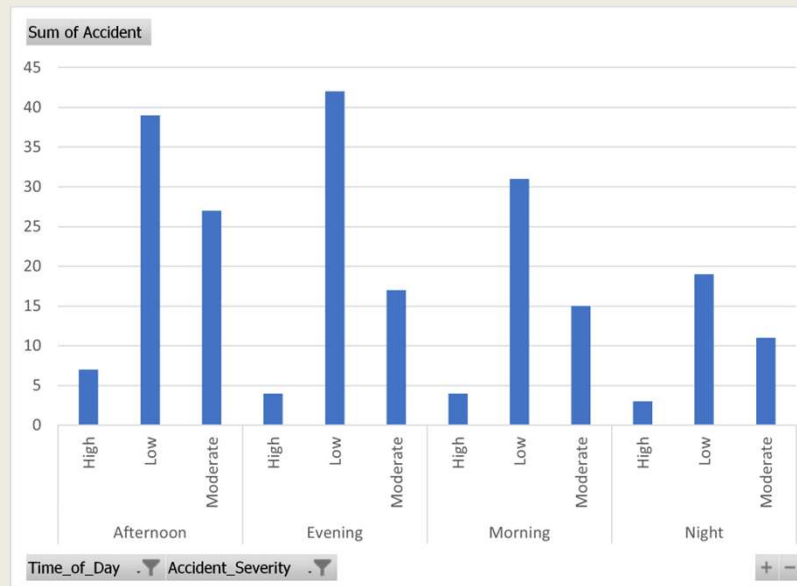
Accidents Vs Weather



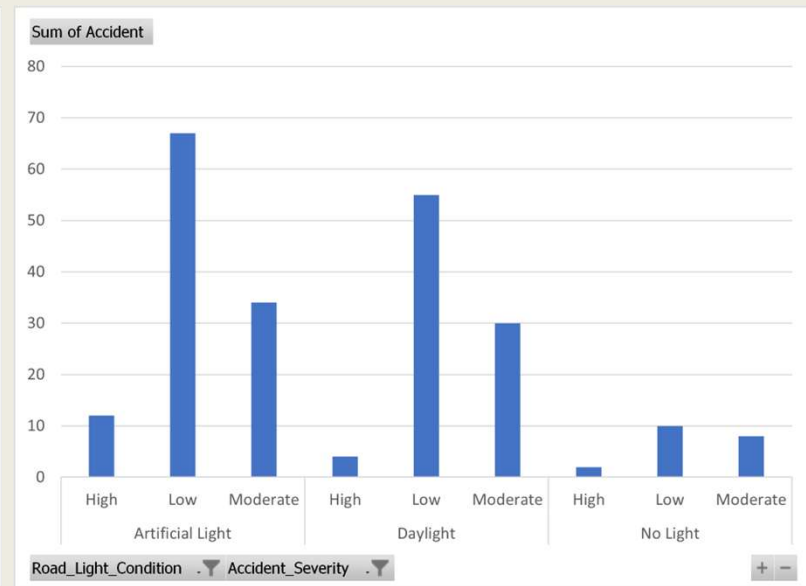
Accidents Vs Road_Type



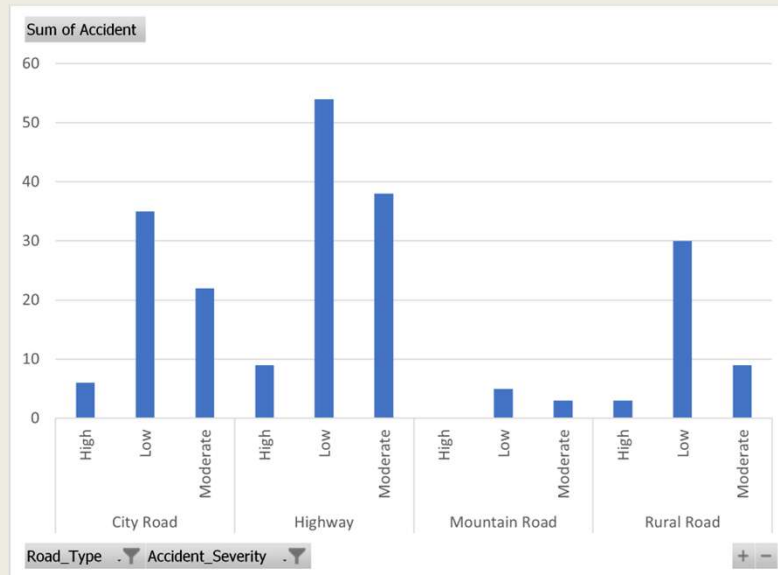
Accidents Vs Accident Severity & Time_of_Day



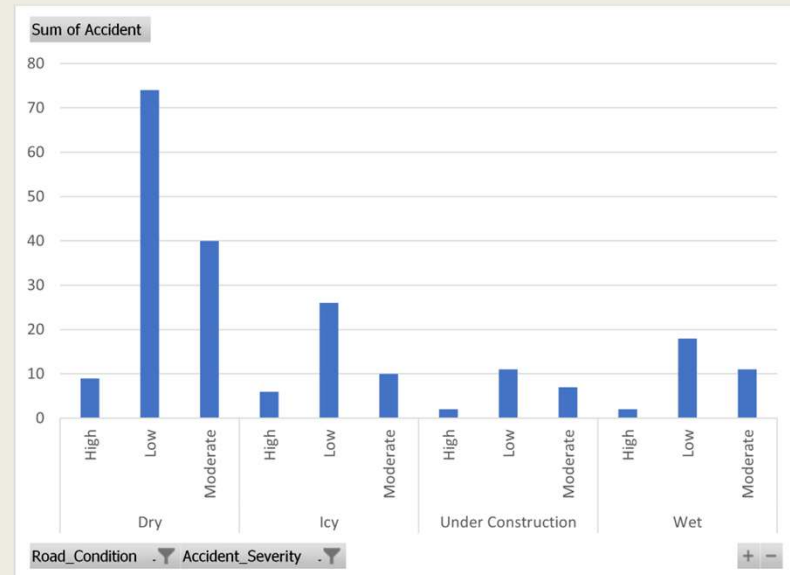
Accidents Vs Accident Severity & Road_Light_Condition



Accidents Vs Accident Severity & Road_Type



Accidents Vs Accident Severity & Road_Condition



1. Importing Required Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

2. Uploading the Dataset

```
uploaded = files.upload()

# Get the name of the uploaded file
file_name = list(uploaded.keys())[0]

# Read the file into a pandas DataFrame
df = pd.read_csv(io.BytesIO(uploaded[file_name]))
```

3. Handling Missing Values

```
# Drop rows with missing target variables
df = df.dropna(subset=["Accident", "Accident_Severity"])

# Fill missing values for numerical columns with their median
numerical_cols = df.select_dtypes(include=["float64"]).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].median())

# Fill missing values for categorical columns with the mode
categorical_cols = df.select_dtypes(include=["object"]).columns
df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode().iloc[0])
```

4. Encoding Categorical Variables

```
# Encode categorical features
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

5. Splitting Features and Target Variables

```
# Split data into features and target variables
X = df.drop(columns=["Accident", "Accident_Severity"])
y_accident = df["Accident"]
y_severity = df["Accident_Severity"]
```

6. Handling Class Imbalance Using SMOTE

```
# Apply SMOTE to balance the datasets
smote = SMOTE(random_state=42, k_neighbors=min(5, df['Accident_Severity'].value_counts().min() - 1)) # Updated SMOTE
X_resampled_accident, y_resampled_accident = smote.fit_resample(X, y_accident)
X_resampled_severity, y_resampled_severity = smote.fit_resample(X, y_severity)
```

7. Splitting Data into Training and Testing Sets

```
# Split into training and testing sets (80-20 split)
X_train_accident, X_test_accident, y_train_accident, y_test_accident = train_test_split(
    X_resampled_accident, y_resampled_accident, test_size=0.2, random_state=42, stratify=y_resampled_accident)
X_train_severity, X_test_severity, y_train_severity, y_test_severity = train_test_split(
    X_resampled_severity, y_resampled_severity, test_size=0.2, random_state=42, stratify=y_resampled_severity)
```

8. Training Random Forest Models

```
# Train Random Forest models for accident occurrence and severity
rf_model_accident = RandomForestClassifier(n_estimators=500, random_state=42, max_depth=20, min_samples_split=3,
                                          min_samples_leaf=1, bootstrap=True, max_features='sqrt',
                                          class_weight='balanced', n_jobs=-1)
rf_model_accident.fit(X_train_accident, y_train_accident)

rf_model_severity = RandomForestClassifier(n_estimators=500, random_state=42, max_depth=20, min_samples_split=3,
                                          min_samples_leaf=1, bootstrap=True, max_features='sqrt',
                                          class_weight='balanced', n_jobs=-1)
rf_model_severity.fit(X_train_severity, y_train_severity)
```

9. Making Predictions

```
# Predictions
y_pred_accident = rf_model_accident.predict(X_test_accident)
y_pred_severity = rf_model_severity.predict(X_test_severity)
```

10. Evaluating Model Performance

```
# Compute accuracy and confusion matrices
accuracy_accident = accuracy_score(y_test_accident, y_pred_accident)
conf_matrix_accident = confusion_matrix(y_test_accident, y_pred_accident)

accuracy_severity = accuracy_score(y_test_severity, y_pred_severity)
conf_matrix_severity = confusion_matrix(y_test_severity, y_pred_severity)
```

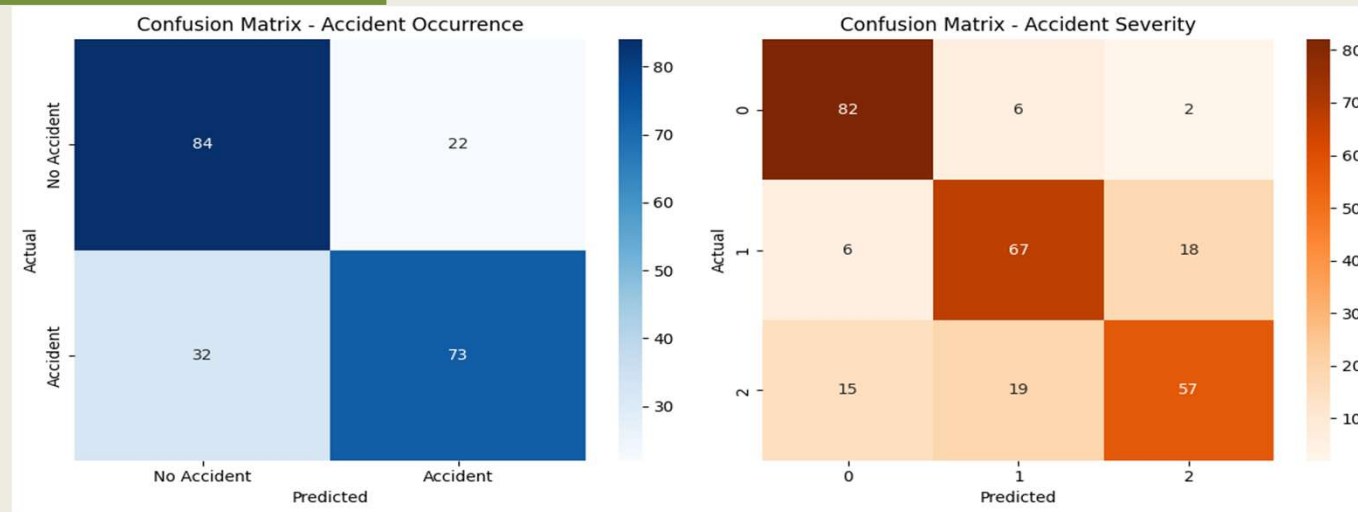
11. Visualizing Results with Heatmaps & Final Output

```
# Plot confusion matrices
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
sns.heatmap(conf_matrix_accident, annot=True, fmt='d', cmap='Blues', ax=axes[0],
            xticklabels=["No Accident", "Accident"], yticklabels=["No Accident", "Accident"])
axes[0].set_xlabel("Predicted")
axes[0].set_ylabel("Actual")
axes[0].set_title("Confusion Matrix - Accident Occurrence")

sns.heatmap(conf_matrix_severity, annot=True, fmt='d', cmap='Oranges', ax=axes[1])
axes[1].set_xlabel("Predicted")
axes[1].set_ylabel("Actual")
axes[1].set_title("Confusion Matrix - Accident Severity")

plt.tight_layout()
plt.show()

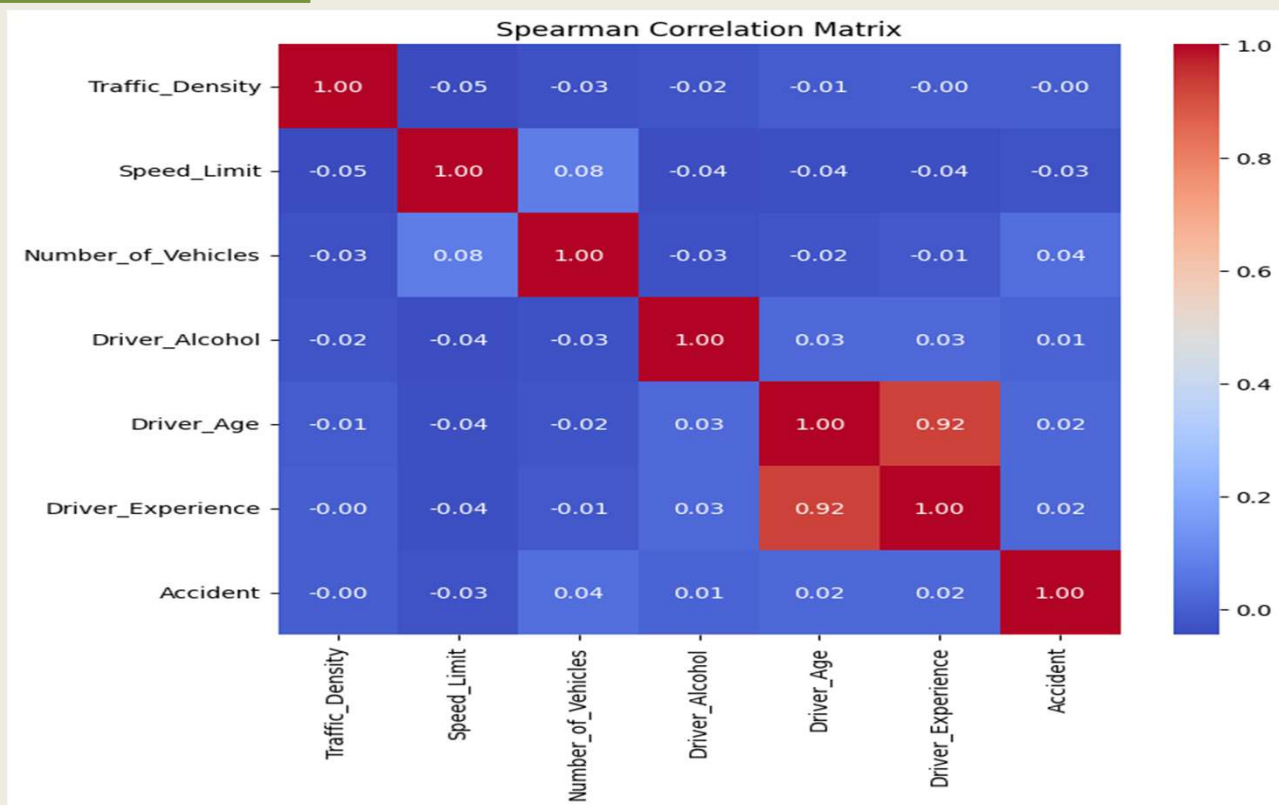
print("Accuracy - Accident Occurrence:", accuracy_accident)
print("Accuracy - Accident Severity:", accuracy_severity)
```

Accuracy - Accident Occurrence: 0.7440758293838863

Accuracy - Accident Severity: 0.7573529411764706

https://colab.research.google.com/drive/1CdiqQ_1bKTMroMYDWBIBHxEh6eyFKzR6?usp=sharing



THANK YOU

