Road Accident Analysis in India

Project Scope:

The main objective of the project is to develop a comprehensive road accident analysis system that can help to identify patterns and trends in road accidents across India. The system should provide a platform for data collection, processing, and analysis to enable policymakers, law enforcement agencies, and other stakeholders to make data-driven decisions and take measures to prevent road accidents and improve road safety.

The project deliverables includes a web-based or mobile application that provides access to real-time road accident data and analysis. The system also includes data processing and analysis modules that can identify patterns and trends in the data and generate reports. Additionally, the system should integrate with hardware tools like vehicle speed detectors and alcohol detectors to identify and alert users about potentially dangerous situations.

The project tasks includes data collection, processing, and analysis; designing and developing the user interface and data visualization tools; integrating hardware tools like vehicle speed detectors and alcohol detectors to identify dangerous situations; developing the SMS notification feature to alert registered mobiles about dangerous situations.

Design Requirements:

Data Collection: The system should be able to collect data from multiple sources, including police reports, hospital records, and traffic cameras. It should also be able to collect data on weather conditions, road infrastructure, and other factors that may contribute to road accidents.

Data Visualization: The system should provide users with data visualization tools that allow them to explore and analyze the data. It should also provide interactive maps that allow users to view the location of accidents and other relevant information.

Hardware Integration: The system should be able to integrate with hardware tools like vehicle speed detectors and alcohol detectors to identify and alert users about potentially dangerous situations.

SMS Notification: The system should be able to send SMS notifications to registered mobiles about potentially dangerous situations, such as when a person is driving under the influence of alcohol.

Data Collection and Processing:

The system collects data from multiple sources, including police reports, hospital records, and traffic cameras. It should also be able to collect data on weather conditions, road infrastructure, and other factors that may contribute to road accidents.

Analysis Techniques:

**Analysing the reasons for road accidents in India:**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Read the csv file into a pandas dataframe

accidents\_df = pd.read\_csv('accident.csv')

# Calculate the number of accidents for each reason

reason\_counts = accidents\_df['Reason'].value\_counts()

# Define the colors for the pie chart using spectral color map

colors = plt.cm.Spectral(np.linspace(0, 1, len(reason\_counts)))

# Plot the data as a pie chart

plt.figure(figsize=(8,8)) # set the size of the figure

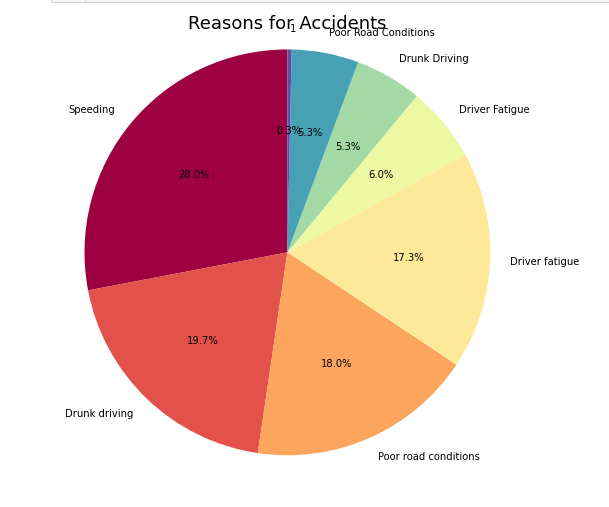
plt.pie(reason\_counts, labels=reason\_counts.index, colors=colors, autopct='%1.1f%%', startangle=90)

plt.title('Reasons for Accidents', fontsize=18)

plt.axis('equal') # make the pie chart circular

plt.show()

**Accident analysis by state:**

We are going to analyze the number of accidents by state to identify which states have a high incidence of accidents and what factors contribute to them. Here is the code to analyze the number of accidents by state:

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.cm as cm

# Load the dataset

accidents\_df = pd.read\_csv('accident.csv')

# Group accidents by state and count them

state\_accidents = accidents\_df.groupby('State')['Accident\_ID'].count().reset\_index()

# Sort states by the number of accidents

sorted\_states = state\_accidents.sort\_values(by='Accident\_ID', ascending=False)

# Display the top 10 states with the most accidents

top\_states = sorted\_states.head(10)

#print(top\_states)

# Define the color map

colors = cm.Spectral(top\_states['Accident\_ID'] / float(max(top\_states['Accident\_ID'])))

# Create a horizontal bar graph with spectral color map

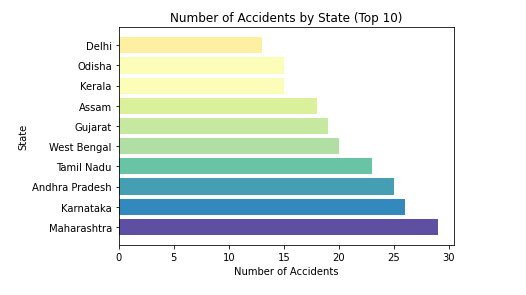
plt.barh(top\_states['State'], top\_states['Accident\_ID'], color=colors)

plt.title('Number of Accidents by State (Top 10)')

plt.xlabel('Number of Accidents')

plt.ylabel('State')

plt.show()



**Analysis of road accidents by weather conditions:**

We can analyze which weather conditions have the highest occurrence of road accidents and investigate how these conditions affect the likelihood of accident.

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.cm as cm

# Load the dataset

accidents\_df = pd.read\_csv('accident.csv')

# Group accidents by weather condition and count them

weather\_accidents = accidents\_df.groupby('Weather\_Conditions')['Accident\_ID'].count().reset\_index()

# Sort weather conditions by the number of accidents

sorted\_weather = weather\_accidents.sort\_values(by='Accident\_ID', ascending=False)

# Display the weather conditions with the most accidents

top\_weather = sorted\_weather.head(10)

#print(top\_weather)

# Define spectral color map

colors = cm.Spectral\_r(top\_weather['Accident\_ID']/float(max(top\_weather['Accident\_ID'])))

# Create a bar graph with spectral color

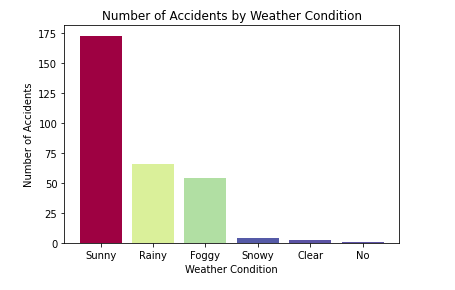
plt.bar(top\_weather['Weather\_Conditions'], top\_weather['Accident\_ID'], color=colors)

plt.title('Number of Accidents by Weather Condition')

plt.xlabel('Weather Condition')

plt.ylabel('Number of Accidents')

plt.show()



**Analysis of the impact of speeding on road accidents:**

We can study the impact of speeding on the frequency and severity of accidents and explore ways to reduce accidents caused by speeding.

import pandas as pd

import matplotlib.pyplot as plt

# Load data into a DataFrame

accidents\_df = pd.read\_csv('accident.csv', usecols=['Speed\_Limit', 'Number\_of\_Deaths'])

# Group data by speed limit and calculate the average number of deaths and injuries

speed\_stats = accidents\_df.groupby('Speed\_Limit', as\_index=False)['Number\_of\_Deaths'].mean()

# Create a line graph to analyze the relationship between speed limit and average number of deaths and injuries

plt.plot(speed\_stats['Speed\_Limit'], speed\_stats['Number\_of\_Deaths'], label='Average Number of Deaths')

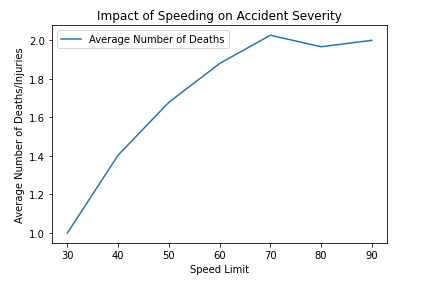
plt.title('Impact of Speeding on Accident Severity')

plt.xlabel('Speed Limit')

plt.ylabel('Average Number of Deaths/Injuries')

plt.legend()

plt.show()



**Analysis of the impact of alcohol on road accidents:**

We can investigate the relationship between alcohol consumption and the likelihood of accidents and explore ways to prevent accidents caused by drunk driving.

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from a CSV file

accidents\_df = pd.read\_csv('accident.csv')

# Filter the data to include only accidents with alcohol involvement

alcohol\_accidents\_df = accidents\_df[accidents\_df['Alcohol\_Involved'] == 'Yes']

# Count the number of alcohol-related accidents in each state

state\_counts = alcohol\_accidents\_df['State'].value\_counts()

# Create a bar plot of the state-wise alcohol-related accidents

plt.bar(state\_counts.index, state\_counts.values, color=plt.cm.Spectral(state\_counts.values/max(state\_counts.values)))

# Set the title and axis labels

plt.title('Alcohol-Related Accidents by State')

plt.xlabel('State')

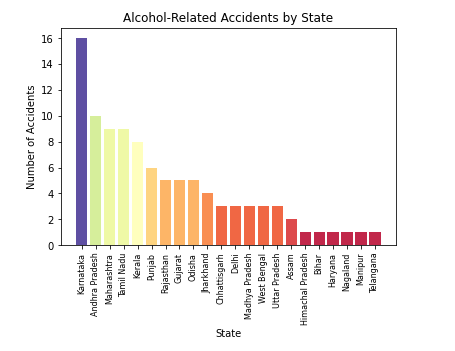
plt.ylabel('Number of Accidents')

# Rotate the x-tick labels and adjust font size

plt.xticks(rotation=90, fontsize=8)

# Display the plot

plt.show()



**Analysis of the impact of rural vs. urban locations on accidents:**

We can investigate whether there is a higher incidence of accidents in rural areas compared to urban areas, and explore the reasons behind this.

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from a CSV file

accidents\_df = pd.read\_csv('accident.csv')

# Create a new column to classify the accidents as rural or urban based on the road type

accidents\_df['Location\_Type'] = accidents\_df['Road\_Type'].apply(lambda x: 'Rural' if x.startswith('R') else 'Urban')

# Count the number of accidents by location type

location\_counts = accidents\_df['Location\_Type'].value\_counts()

colors = ['#ff7f0e', '#1f77b4']

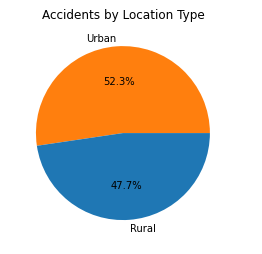
plt.pie(location\_counts.values, labels=location\_counts.index, colors=colors, autopct='%1.1f%%')

# Set the title and axis labels

plt.title('Accidents by Location Type')

# Display the plot

plt.show()



**Sending SMS Alert to Family Members:**

When integrated with hardware , takes input from alcohol detector whether the user is drunk or not and continuously monitors the speed of the vehicle and takes it as input.

It also requests to enter the captcha , if entered wrong 3 times in a row, then treats as drunk and send SMS alert to the registered mobiles.

This code uses Python library Twilio to communicate with external systems such as SMS gateways or APIs to send alerts to registered mobiles about potentially dangerous situations.

import os

import secrets

import time

from twilio.rest import Client

# Set environment variables for your credentials

# Read more at http://twil.io/secure

account\_sid = "AC00709cd49b7b8b39ff08b38420b6508a"

auth\_token = "6f4b2d849b1a7787da5b687fe17e66bb"

client = Client(account\_sid, auth\_token)

# Define the phone numbers

from\_number = "+16205071602"

to\_number = "+919492333588"

# Get the user input for alcohol consumption

alcohol\_input = input("Have you consumed alcohol? (yes/no) ").lower()

if alcohol\_input == "yes":

    alcohol\_detected = True

else:

    alcohol\_detected = False

    # Generate a secure random key with a given length

    def generate\_key(length):

        return secrets.token\_hex(length//2)

    # Generate a captcha and return the key

    def generate\_captcha():

        # Generate a 6-character captcha

        captcha = generate\_key(6)

        print(f"Captcha: {captcha}")

        return captcha

    # Check if the user's answer matches the key

    def check\_answer(answer, key):

        return answer == key

    # Set the number of allowed attempts

    max\_attempts = 3

    attempts = 0

    while attempts < max\_attempts:

        # Generate a captcha and prompt the user to enter an answer

        captcha\_key = generate\_captcha()

        user\_answer = input("Enter the captcha: ")

        # Check if the user's answer matches the key

        if check\_answer(user\_answer, captcha\_key):

            print("Success! Captcha matches the key.")

            break

        else:

            attempts += 1

            if attempts == max\_attempts:

                print("Error! Three wrong attempts, driver has consumed alcohol.")

                alcohol\_detected = True

            else:

                print(f"Error! Captcha does not match the key. {max\_attempts - attempts} attempts remaining.")

# Get the user input for speed

speed\_input = input("Enter your current speed (km/h): ")

speed = int(speed\_input)

# Check if speed is above the limit and alcohol is detected

if speed > 100 :

    # Construct the message to be sent

    message\_body = f"Your Companion is overspeeding and Speed is {speed\_input} km/h."

    # Send the message

    message = client.messages.create(

        body=message\_body,

        from\_=from\_number,

        to=to\_number

    )

if alcohol\_detected:

    # Construct the message to be sent

    message\_body = f"Your Companion has consumed alcohol and is trying to drive the car. Please send emergency services."

    # Send the message

    message = client.messages.create(

        body=message\_body,

        from\_=from\_number,

        to=to\_number

    )

if speed > 100 and alcohol\_detected:

    # Construct the message to be sent

    message\_body = f"Your Companion is overspeeding and Speed is {speed\_input} km/h and has consumed alcohol. Please send emergency services."

    # Send the message

    message = client.messages.create(

        body=message\_body,

        from\_=from\_number,

        to=to\_number

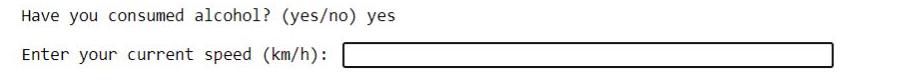
    )

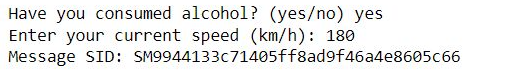
    # Print the message SID for confirmation

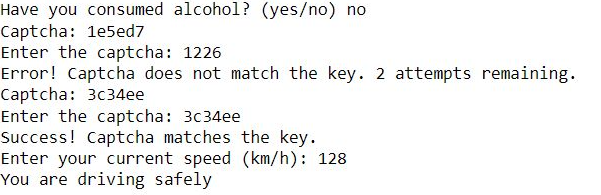
    print("Message SID:", message.sid)

else:

    print("You are driving safely")

User Interface and Outputs:





Testing and Validation:

