# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score, roc\_curve

!pip install streamlit

!pip install pyngrok

from sklearn.preprocessing import LabelEncoder

from google.colab import files

uploaded=files.upload()

# Load dataset

df = pd.read\_csv("RTA Dataset.csv")

df.head()



# Preprocessing and Exploration

f = df.drop(columns=['ID'], errors='ignore')

# Handle missing values

df = df.dropna()

df

df.head(10)

df.columns

df.info()

df['Time']=pd.to\_datetime(df['Time'])

df.loc[55:60,['Road\_surface\_conditions']]

df.describe()

df.duplicated()

df.duplicated().sum()

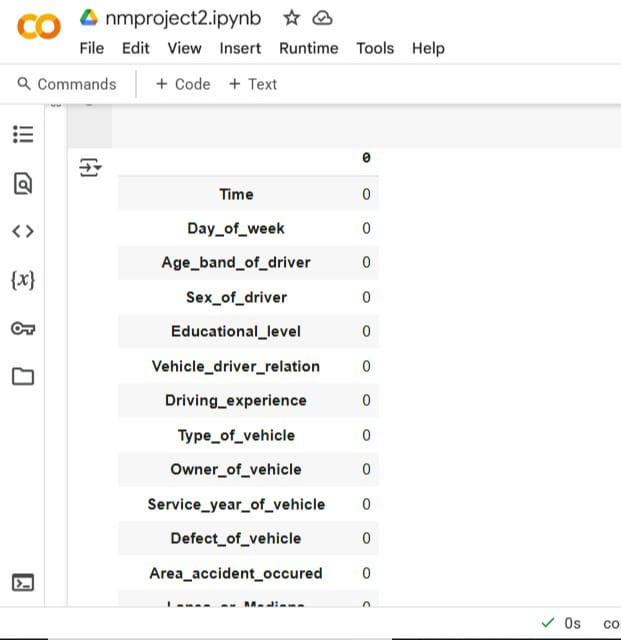
df.groupby('Accident\_severity').size()

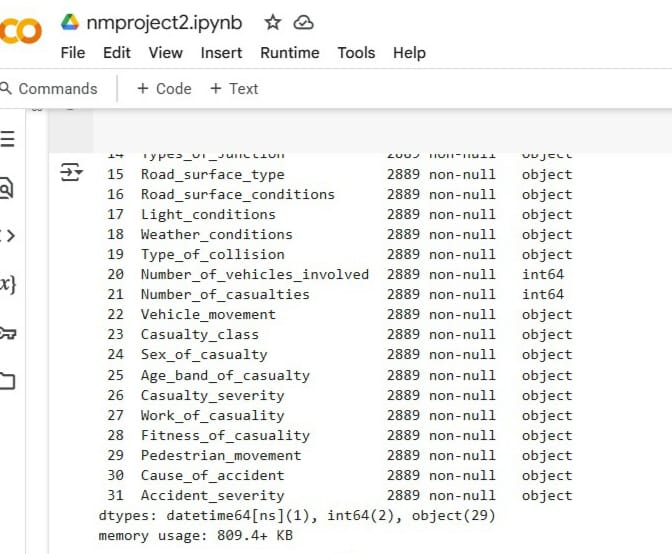
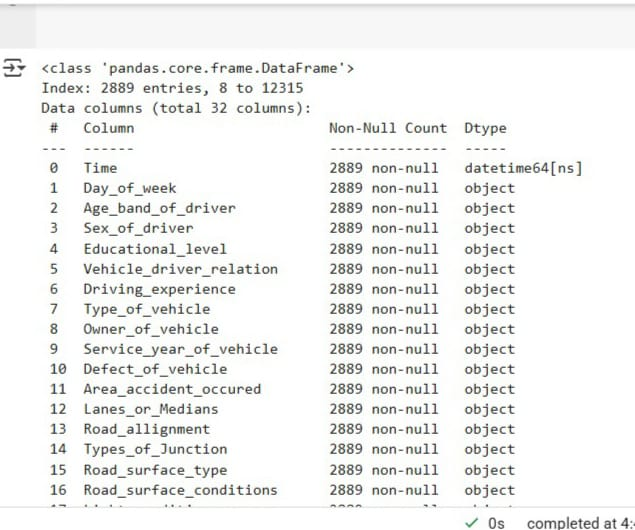
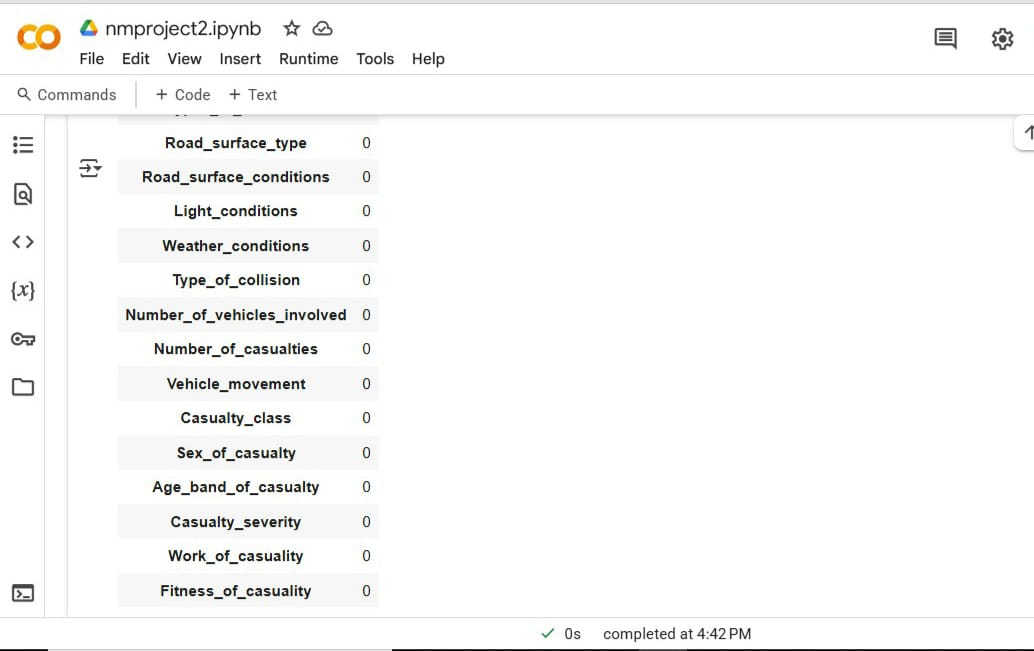
df['Cause\_of\_accident'].value\_counts()

df.isnull()

df.isnull().any()

df.isnull().sum()





# Data Visualisation and EDA

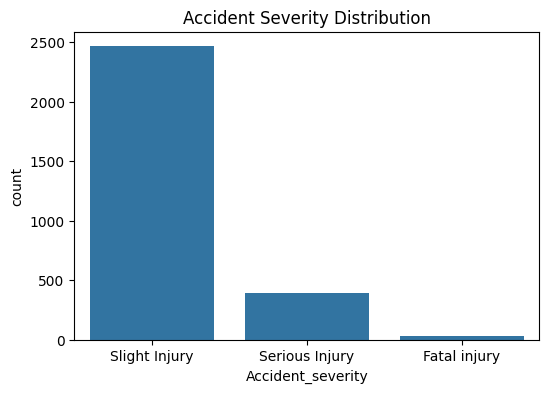
# Accident severity distribution

plt.figure(figsize=(6,4))

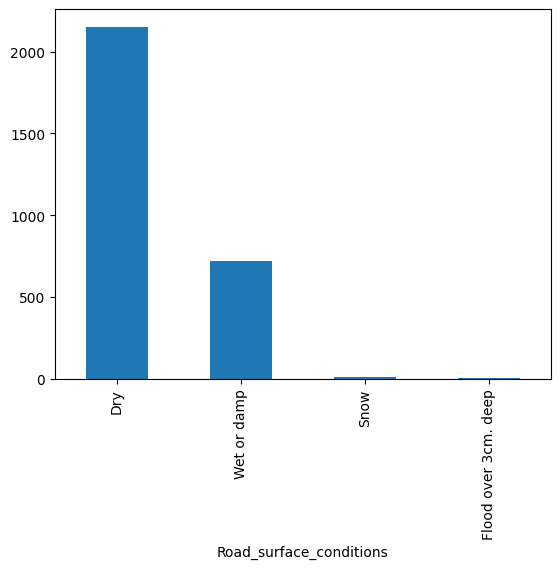
sns.countplot(x='Accident\_severity', data=df)

plt.title("Accident Severity Distribution")

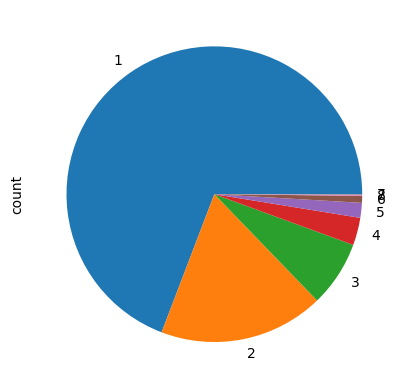
plt.show()



df.Road\_surface\_conditions.value\_counts().plot(kind='bar')



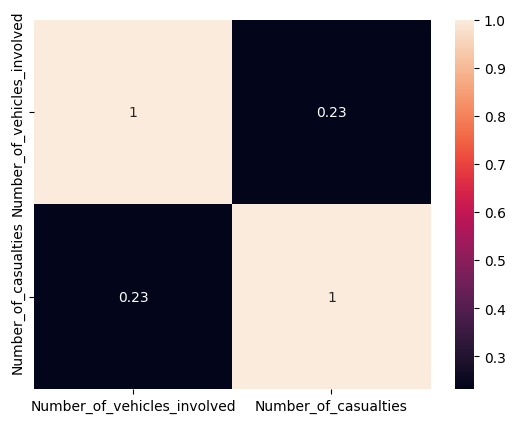
df.Number\_of\_casualties.value\_counts().plot(kind='pie')



correlation\_matrix = df[['Number\_of\_vehicles\_involved','Number\_of\_casualties']].corr()

sns.heatmap(correlation\_matrix,annot=True)

plt.show()



df['hour']=pd.to\_datetime(df['Time']).dt.hour

df['minute']=pd.to\_datetime(df['Time']).dt.minute

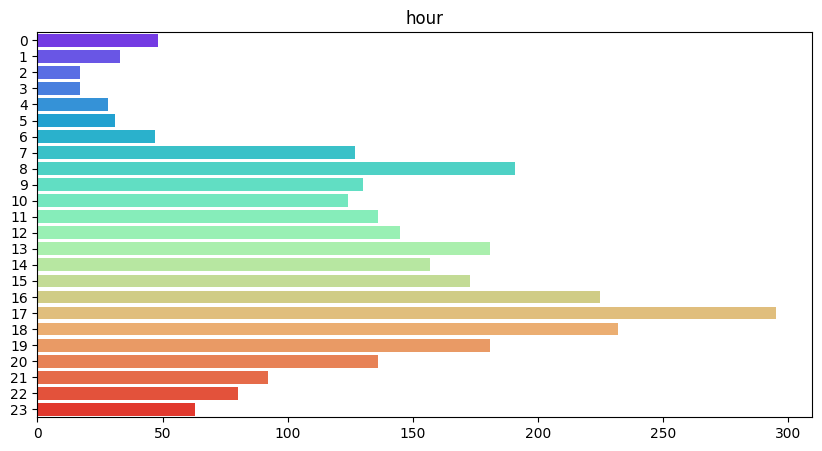
plt.figure(figsize=(10,5))

sns.countplot(df,y='hour',palette='rainbow')

plt.xlabel('')

plt.ylabel('')

plt.title('hour')



# Feature Engineering

le = LabelEncoder()

for col in df.columns:

if df[col].dtype == 'object':

df[col] = le.fit\_transform(df[col])

correlations = df.corr()['Accident\_severity'].abs().sort\_values(ascending=False)

top\_features = correlations[1:11]

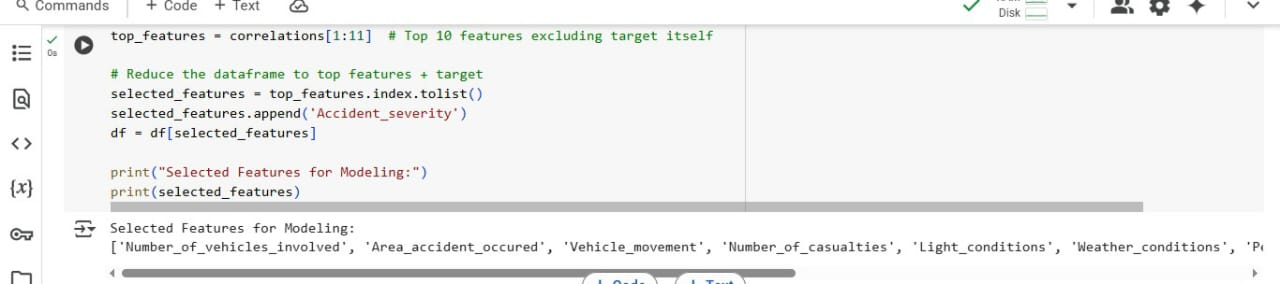
selected\_features = top\_features.index.tolist()

selected\_features.append('Accident\_severity')

df = df[selected\_features]

print("Selected Features for Modeling:")

print(selected\_features)



# Model Building

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score, f1\_score

# Define models

models = {

"Logistic Regression": LogisticRegression(max\_iter=200),

"Decision Tree": DecisionTreeClassifier(),

"Random Forest": RandomForestClassifier(),

"XGBoost": XGBClassifier()

}

# Train and evaluate each model

results = []

x=df.drop('Accident\_severity',axis=1)

y=df['Accident\_severity']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred, average='weighted')

results.append({

'Model': name,

'Accuracy': round(acc, 4),

'F1 Score': round(f1, 4)

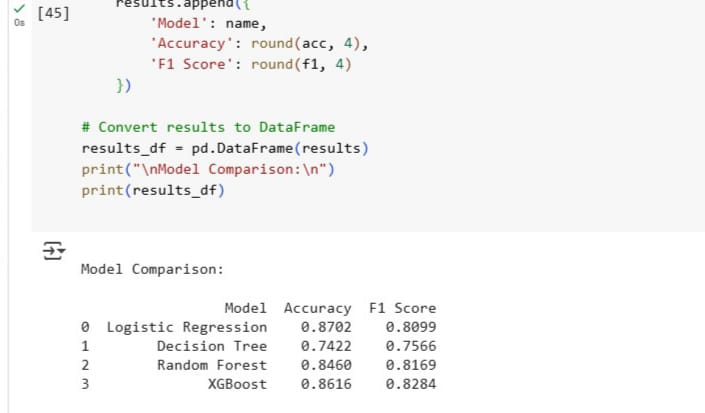
})

# Convert results to DataFrame

results\_df = pd.DataFrame(results)

print("\nModel Comparison:\n")

print(results\_df)



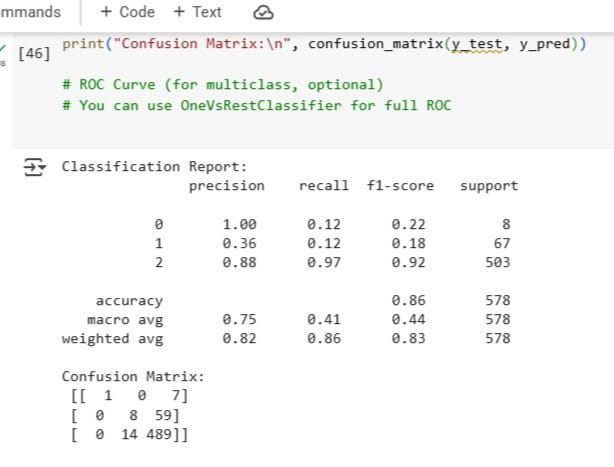
# Evaluation

y\_pred = model.predict(X\_test)

# Metrics

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))



# Deployment

%%writefile app1.py

import streamlit as st

import numpy as np

import pickle

st.title("🚦 Traffic Accident Severity Prediction")

# Load trained model

model = pickle.load(open('model.pkl', 'rb'))

# Get user input

age = st.selectbox("Age Band of Driver", ['Under 18', '18-30', '31-50', 'Above 51'])

sex = st.selectbox("Sex of Driver", ['Male', 'Female'])

day = st.selectbox("Day of Week", ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'])

weather = st.selectbox("Weather Condition", ['Clear', 'Raining', 'Fog', 'Windy'])

# Encode inputs manually (must match training)

age\_map = {'Under 18': 0, '18-30': 1, '31-50': 2, 'Above 51': 3}

sex\_map = {'Male': 0, 'Female': 1}

day\_map = {'Sunday': 0, 'Monday': 1, 'Tuesday': 2, 'Wednesday': 3, 'Thursday': 4, 'Friday': 5, 'Saturday': 6}

weather\_map = {'Clear': 0, 'Raining': 1, 'Fog': 2, 'Windy': 3}

# Create input array with dummy values for missing features

input\_data = np.array([[

age\_map[age],

sex\_map[sex],

day\_map[day],

weather\_map[weather],

0, 1, 0, 1, 2, 1 # Dummy values for missing 6 features

]])

if st.button("Predict"):

prediction = model.predict(input\_data)

severity = ['Fatal Injury', 'Serious Injury', 'Slight Injury']

st.success(f"Predicted Accident Severity: {severity[prediction[0]]}")

from pyngrok import ngrok

# Kill existing tunnels

ngrok.kill()

# Start Streamlit in background

# get\_ipython().system\_raw('streamlit run app.py &')

# Connect to ngrok

public\_url = ngrok.connect(port='8501')

print("Streamlit App URL:", public\_url)

import pickle

pickle.dump(model, open('model.pkl', 'wb'))

!streamlit run app.py &

