**Phase-3**

**Student Name:** J HARSHINI

**Register Number:** 410723104023

**Institution:** DHANALAKSHMI COLLEGE OF ENGINEERING

**Department:** COMPUTER SCIENCE AND ENGINEERING

**Date of Submission:** 14-05-2025

**Github Repository Link:**

[**https://github.com/Harshini-jk/NM\_Harshini**](https://github.com/Harshini-jk/NM_Harshini)

### **1. Problem Statement**

*Enhancing road safety with AI-driven traffic accident analysis and prediction -- With the continuous rise in urbanization and vehicle usage, road traffic accidents remain a leading cause of injury and death globally. The need for innovative, data-driven solutions to mitigate these incidents is more pressing than ever. This project aims to leverage Artificial Intelligence (AI) and Machine Learning (ML) techniques to analyze historical traffic data, identify high-risk zones and conditions, and develop predictive models that can forecast the likelihood of accidents before they occur.*

### **2. Abstract**

*This study leverages artificial intelligence (AI) to analyze and predict traffic accidents, aiming to enhance road safety. By integrating machine learning algorithms with comprehensive traffic data, we identify high-risk areas, contributing factors, and predictive patterns. Our AI-driven approach enables proactive measures to mitigate accidents, informing policy decisions and targeted interventions. Results show significant potential for reducing accidents and improving road safety. This research demonstrates the effectiveness of AI in transforming transportation systems and saving lives.*

### **3. System Requirements:**

***Hardware:***

*• Minimum 8 GB RAM*

*• Intel i5 processor or higher*

***Software:***

*• Python 3.8+*

*• Jupyter Notebook / Google Colab*

*• Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost.*

### **4. Objectives:**

*To develop and implement an AI-driven system for traffic accident analysis and prediction, aiming to:*

*1. Identify high-risk accident zones and contributing factors*

*2. Predict potential accident hotspots and times*

*3. Inform data-driven decision-making for road safety interventions*

*4. Reduce traffic accidents and enhance overall road safety*

**5.flowchart of project work flow:**

Data Collection

Preprocessing

Exploratory data analysis

Feature Engineering

Model Training

Evaluation

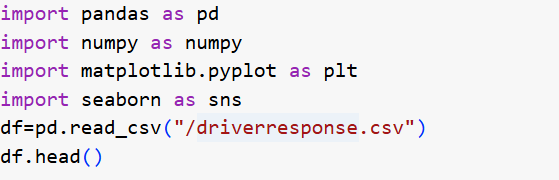
Deployment

### **6. Dataset Description:**

source: Kaggle – driver response

Type: Public

Size:5\*21 columns

****

### 

### **7. Data Preprocessing:**

1.Handling missing values

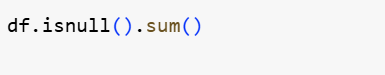
2. Data normalization

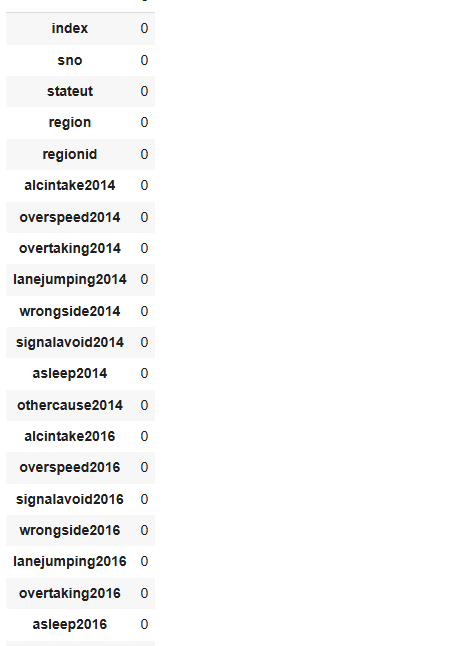
3. Feature engineering (extracting relevant features)

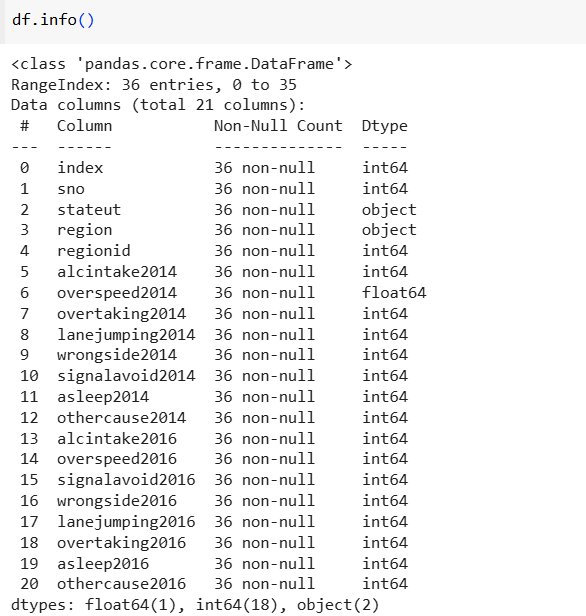
4. Encoding categorical variables

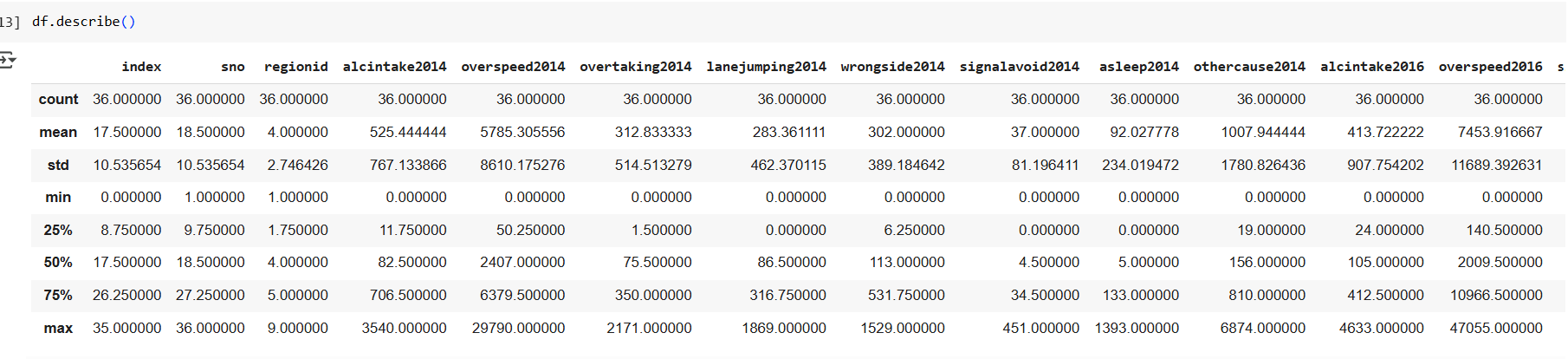
5. Data transformation (converting data types)

6. Removing duplicates and outliers.

**

**

**

**

**

**8. Exploratory Data Analysis (EDA):**

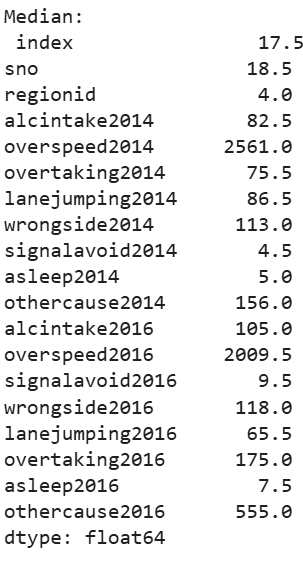
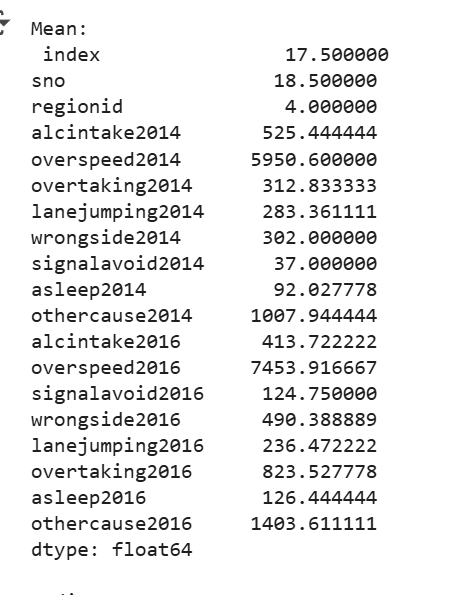
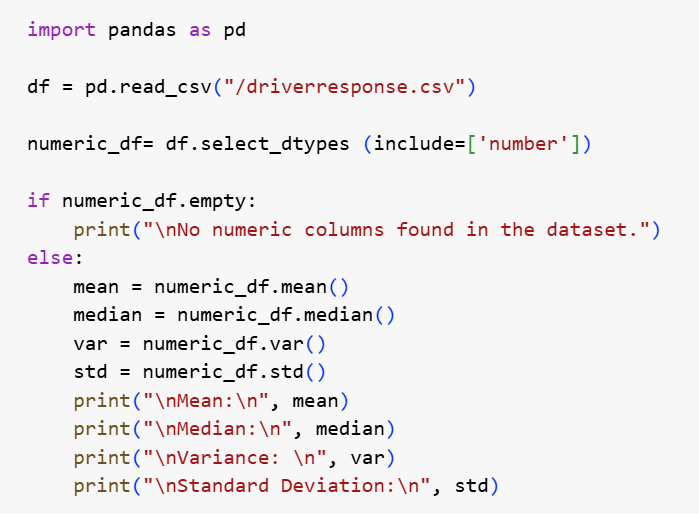
*1. Data cleaning and preprocessing*

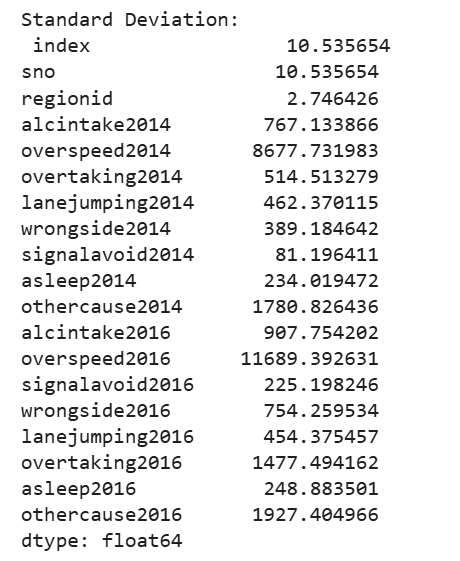
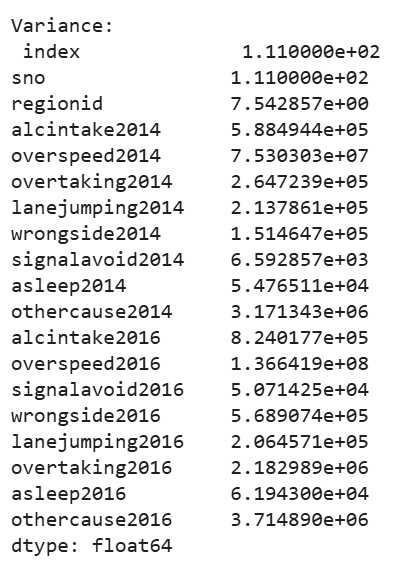
*2. Summary statistics (mean, median, mode, etc.)*

*3. Data visualization (plots, charts, etc.)*

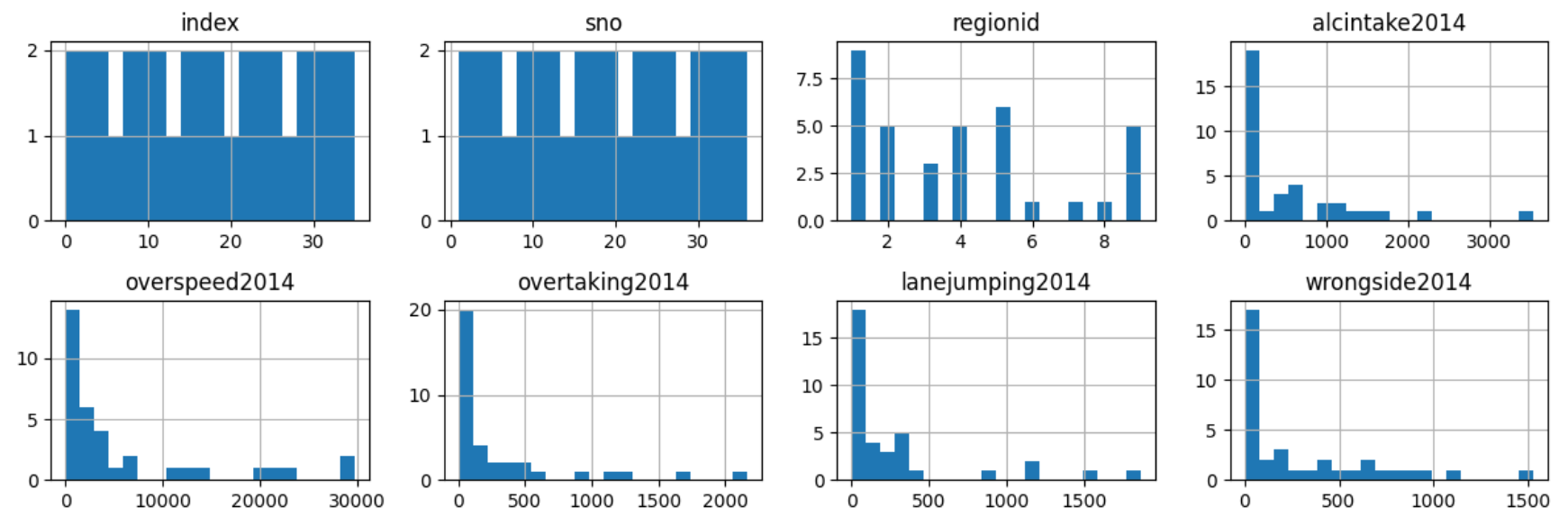
*4. Correlation analysis*

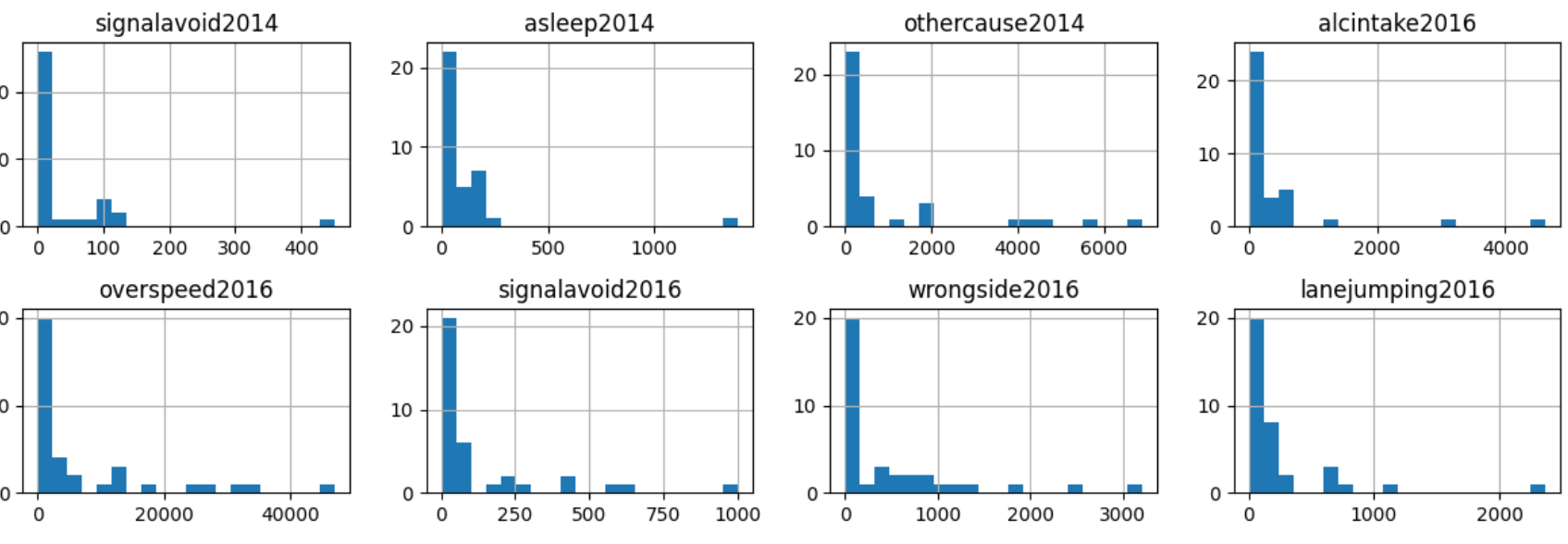
*5. Pattern identification*

**

**

### 





### **9. Feature Engineering:**

*1. Extract time-based features (hour, day, month)*

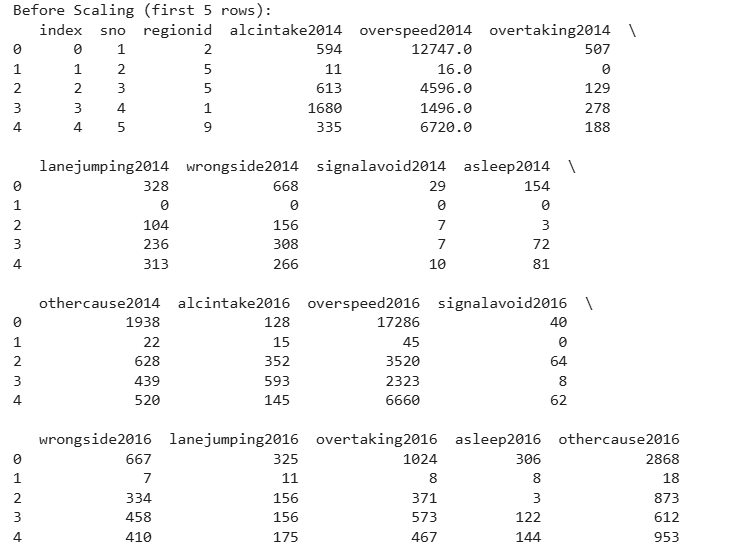
*2. Create location-based features (latitude, longitude, proximity to intersections)*

*3. Calculate traffic-related features (traffic volume, speed)*

*4. Encode categorical variables (weather, road conditions)*

*5. Derive accident severity features (injury/fatality rates)*

### 

**

### 

### **10. Model Building**

*1. Select algorithms: Choose suitable machine learning algorithms (e.g., logistic regression, decision trees, random forest, neural networks)*

*2. Train models: Train models using the training dataset*

*3. Hyperparameter tuning: Optimize model hyperparameters for better performance*

*4. Model evaluation: Evaluate model performance using metrics (accuracy, precision, recall, F1-score)*

*5. Model selection: Select the best-performing model for deployment.*

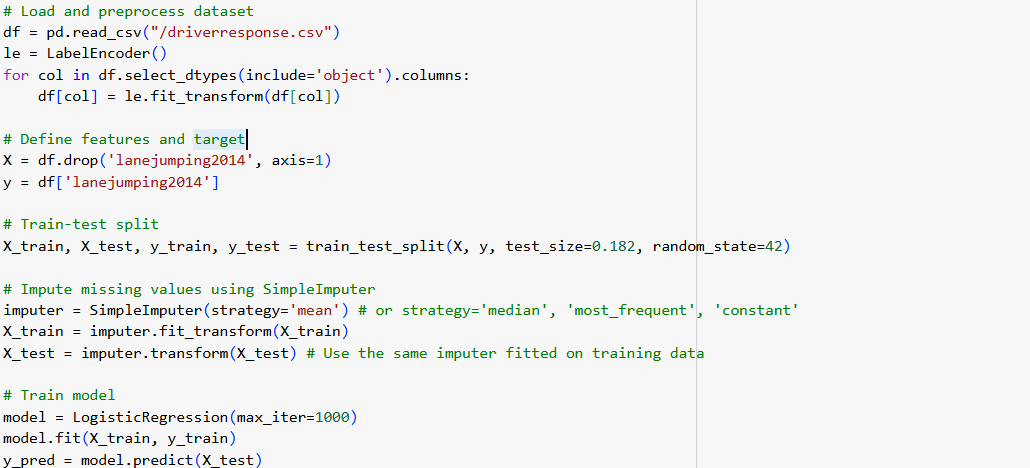
### 

### **11.**

### **11.Model Evaluation**

*1.SCIKIT - learn\*: Model evaluation metrics (Python)*

*2. Metrics: Accuracy, precision, recall, F1-score*

**

### **12. Deployment**

*Deployment Plan:\**

*1. Integration with Existing Infrastructure: Integrate the AI system with existing traffic management systems, surveillance cameras, and sensors.*

*2. Real-time Data Feed: Establish a real-time data feed from various sources, including traffic cameras, sensors, and emergency services.*

*3. Cloud-based Deployment: Deploy the system on a cloud-based platform for scalability, reliability, and maintenance.*

**13. Source code**  
  
*import pandas as pd*

*import numpy as numpy*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*df=pd.read\_csv("/driverresponse.csv")*

*df.head()*

*df.isnull().sum()*

*df.fillna(0,inplace=True)*

*df.isnull().sum()*

*df.info()*

*df.describe()*

*print(df.duplicated().sum())*

*print("Original shape:",df.shape)*

*numeric\_df= df.select\_dtypes (include=['number'])*

*if numeric\_df.empty:*

*print("\nNo numeric columns found in the dataset.")*

*else:*

*mean = numeric\_df.mean()*

*median = numeric\_df.median()*

*var = numeric\_df.var()*

*std = numeric\_df.std()*

*print("\nMean:\n", mean)*

*print("\nMedian:\n", median)*

*print("\nVariance: \n", var)*

*print("\nStandard Deviation:\n", std)*

*for col in df.columns:*

*print(f"{col}: {df[col].nunique()} unique values")*

*for col in df.select\_dtypes(include='object').columns:*

*print(f"\n{col} value counts:")*

*print(df[col].value\_counts())*

*print(df.shape)*

*numeric\_cols = df.select\_dtypes(include=['float64', 'int64']).columns*

*scaler = StandardScaler()*

*df\_scaled = df.copy()*

*df\_scaled[numeric\_cols] = scaler.fit\_transform(df[numeric\_cols])*

*print("\nBefore Scaling (first 5 rows):")*

*print(df[numeric\_cols].head())*

*print("\nAfter Standardization (first 5 rows):")*

*print(df\_scaled[numeric\_cols].head())*

*df.hist(bins=20, figsize=(12, 10))*

*plt.tight\_layout()*

*plt.show()*

*le = LabelEncoder()*

*for col in df.select\_dtypes(include='object').columns:*

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

*df = df.dropna()*

*X = df.drop('lanejumping2014', axis=1)  # Drop the target column from features*

*y = df['lanejumping2014']  # This is the target column*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.182, random\_state=42)*

*rf = RandomForestClassifier(n\_estimators=100, random\_state=42, class\_weight='balanced')*

*rf.fit(X\_train, y\_train)*

*y\_pred = rf.predict(X\_test)*

*class\_labels = sorted(y.unique())*

*print("Accuracy:", accuracy\_score(y\_test, y\_pred))*

*print("\nClassification Report: \n", classification\_report(y\_test, y\_pred, labels=class\_labels, zero\_division=1))*

*from sklearn.preprocessing import LabelEncoder*

*le = LabelEncoder()*

*for col in df.select\_dtypes(include='object').columns:*

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

*X = df.drop('lanejumping2014', axis=1)  # Drop the target column from features*

*y = df['lanejumping2014']  # This is the target column*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.182, random\_state=42)*

*rf = RandomForestClassifier(n\_estimators=100, random\_state=42)*

*rf.fit(X\_train, y\_train)*

*y\_pred = rf.predict(X\_test)*

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

*f1 = f1\_score(y\_test, y\_pred, average='weighted')  # Use 'weighted' or another suitable average*

*rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))*

*if len(rf.classes\_) == 2:*

*y\_prob = rf.predict\_proba(X\_test)[:, 1]*

*fpr, tpr, thresholds = roc\_curve(y\_test, y\_prob)*

*auc\_score = auc(fpr, tpr)*

*else:*

*auc\_score = "N/A"  # Not applicable for multi-class problems*

*cm = confusion\_matrix(y\_test, y\_pred)*

*plt.figure(figsize=(8,6))*

*sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=rf.classes\_, yticklabels=rf.classes\_)*

*plt.xlabel('Predicted')*

*plt.ylabel('Actual')*

*plt.title('Confusion Matrix')*

*plt.show()*

*if len(rf.classes\_) == 2:*

*plt.figure(figsize=(8,6))*

*plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {auc\_score:.2f})')*

*plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')*

*plt.xlim([0.0, 1.0])*

*plt.ylim([0.0, 1.05])*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('Receiver Operating Characteristic (ROC) Curve')*

*plt.legend(loc='lower right')*

*plt.show()*

*cv\_scores = cross\_val\_score(rf, X, y, cv=5)*

*evaluation\_results = {*

*'Metric': ['Accuracy', 'F1 Score', 'ROC AUC', 'RMSE'],*

*'Value': [accuracy, f1, auc\_score, rmse]*

*}*

*evaluation\_df = pd.DataFrame(evaluation\_results)*

*print("Evaluation Metrics and Model Comparison:")*

*print(evaluation\_df)*

*print(f"\nCross-Validation Scores: {cv\_scores}")*

*print(f"Mean Cross-Validation Score: {cv\_scores.mean()}")*

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

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.preprocessing import LabelEncoder*

*from sklearn.impute import SimpleImputer  # Import SimpleImputer*

*df = pd.read\_csv("/driverresponse.csv")*

*le = LabelEncoder()*

*for col in df.select\_dtypes(include='object').columns:*

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

*X = df.drop('lanejumping2014', axis=1)*

*y = df['lanejumping2014']*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.182, random\_state=42)*

*imputer = SimpleImputer(strategy='mean') # or strategy='median', 'most\_frequent', 'constant'*

*X\_train = imputer.fit\_transform(X\_train)*

*X\_test = imputer.transform(X\_test) # Use the same imputer fitted on training data*

*model = LogisticRegression(max\_iter=1000)*

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

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

*comparison\_df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})*

*comparison\_df = comparison\_df.groupby(comparison\_df.index).first() # Ensure index is unique*

*labels = comparison\_df.index.astype(str)*

*x = range(len(labels))*

*plt.figure(figsize=(10, 6))*

*plt.bar(x, comparison\_df['Actual'], width=0.4, label='Actual', align='center', color='skyblue')*

*plt.bar([i + 0.4 for i in x], comparison\_df['Predicted'], width=0.4, label='Predicted', align='center', color='salmon')*

*plt.xticks([i + 0.2 for i in x], labels)*

*plt.xlabel('Class Labels (lanejumping2014)')*

*plt.ylabel('Count')*

*plt.title('Actual vs Predicted Counts (Bar Graph)')*

*plt.legend()*

*plt.grid(axis='y', linestyle='--', alpha=0.7)*

*plt.tight\_layout()*

*plt.show()*

**14. Future scope**

*1. Reduced Accidents: Further reduction in traffic accidents and fatalities.*

*2. Improved Traffic Flow: Enhanced traffic flow and reduced congestion.*

*3. Data-Driven Policy Making: Informed policy decisions for urban planning, transportation, and safety.*

*4. Increased Efficiency: Improved emergency response times and resource allocation.*

**15. Team Members and Roles:**

**Google colab:** [**https://colab.research.google.com/drive/1zC24KaQD9n4DAM6uyFrZ7HA\_BVW3G7uo?authuser=1&pli=1**](file:///C:\Users\janub\AppData\Local\Microsoft\Windows\INetCache\IE\NPG4WJGS\__%7bEDC978D6-4D53-4B2F-A265-5805674BE568%7d_Phase-3(Harshini%20J)%5b1%5d_docx_thumb%5b1%5d.png)**#**

|  |  |  |  |
| --- | --- | --- | --- |
| SN NO | NAMES | ROLES | RESPONSIBILITY |
| 1. | DEEPALAKSHMI.A | TEAM MEMBER | DATA COLLECTION &DATA PREPROCESSING |
| 2. | ISHWARYA.S | TEAM MEMBER | EXPLORATORY DATA ANALYSIS &FEATURE ENGINEERING |
| 3. | HARSHINI.J | TEAM MEMBER | MODEL BUILDING &MODEL EVALUATION |
| 4. | LOGAPRIYA | TEAM LEADER | VISUALISATION INTERPRETION,DEPLOYMENT |