PROJECT REPORT

Road Safety and Accident Risk Analysis

(Exploratory Data Analysis and Clustering of US Traffic Accident Data)

Submitted for the partial fulfilment of the requirement for the Degree in

BACHELOR OF COMPUTER APPLICATIONS

Submitted By

Sakshi Rawat, University Roll No: 2221887

Under the guidance of:

Dr. Anupriya

Assistant Professor

SCHOOL Of COMPUTING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

JUNE,2025

CERTIFICATE

This is to certify that the project titled **"Road Safety and Accident Risk Analysis" submitted** by **Sakshi Rawat**, to Graphic Era Hill University for the award of the degree of **Bachelor of Computer Application**, is a bona fide record of the research work done by her under our supervision. The contents of this project in full or in parts have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Anupriya
Project Guide
Assistant Professor
GEHU, Dehradun

Place: Dehradun
Date: 05/06/2025

ACKNOWLEDGEMENT

I take this opportunity to express my deep sense of gratitude and respectful regards to my guide, Dr. Anupriya, at GEHU, Dehradun, for her valuable suggestions, supervision, and amiable guidance.

I cannot fully express my appreciation for her constant and untiring support, as well as her stimulating guidance throughout the session.

I also extend my sincere thanks to all the respected faculty members of the School of Computing for their constant encouragement and support.

Sakshi Rawat

Roll no. 2221887

ABSTRACT

This project explores traffic accident data across the United States with the objective of identifying patterns and clusters of states that share similar profiles regarding fatal collisions. The analysis uses a dataset provided by FiveThirtyEight, originally compiled from the National Highway Traffic Safety Administration and the National Association of Insurance Commissioners. A comprehensive exploratory data analysis was conducted using Python and Streamlit to build an interactive web application. Key steps included data wrangling, visualization of pairwise relationships, and the application of statistical and machine learning methods.

The project started by presenting the raw data and providing a descriptive statistical summary, along with visualizations such as pairwise scatter plots and correlation matrices. A multivariate linear regression model was implemented to assess the influence of factors like speeding, alcohol consumption, and first-time involvement in accidents on the fatal collision rate. Dimensionality reduction using Principal Component Analysis (PCA) was performed on standardized data to better understand underlying structures and variance distribution.

Clustering using KMeans was employed to group states with similar traffic accident profiles, supported by scree plots to determine the optimal number of clusters. The results were visualized both in PCA scatter plots and violin plots to interpret feature differences between clusters. Additionally, miles driven data was incorporated to calculate the total number of fatal collisions per cluster, allowing prioritization of intervention efforts.

An extended analysis included insurance-related features (premiums and losses) to examine their relationship with fatal collisions, utilizing PCA, KMeans, and linear regression models. The Streamlit application serves as a dynamic and interactive platform for data upload, exploration, and visualization, enabling policymakers and researchers to make informed decisions about targeted interventions to reduce traffic fatalities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The website developed in this project serves as an interactive data visualization tool designed to analyze traffic accident data. By leveraging modern web technologies and interactive dashboards, the platform enables users to explore, filter, and visualize accident data efficiently. The website aims to simplify complex datasets for a wide range of users, including policymakers, researchers, and the general public, making the analysis process intuitive and accessible.

1.2 Problem Definition

Analyzing large traffic accident datasets is time-consuming and often requires programming skills, making it inaccessible to many users. Existing solutions usually lack interactivity and intuitive features that help users explore data effectively. This project addresses the need for a simpler, more user-friendly way to analyze traffic accident data.

1.3 Aim of the Project

The aim is to build an interactive, web-based tool for analyzing accident data, enabling users to explore trends and statistics without needing advanced technical skills.

1.4 Objectives

This project aims to import and preprocess data, build an interactive interface, and provide dynamic visualizations. Users can filter data by date, location, and severity, supporting precise analysis.

1.5 Goals

The project seeks to create a fast, responsive web app that works across devices. It will help users visualize trends, identify high-risk areas, and make informed decisions based on clear visualizations.

1.6 Need

There’s a need for an accessible tool to simplify traffic accident data analysis. Current tools often require technical knowledge or lack interactivity. This project fills that gap with an intuitive, user-friendly platform.

1.7 Features to Highlight

Features include interactive dashboards with dynamic filters, responsive design, and engaging visualizations. Users can export data and download reports for further analysis.

1.8 Benefits

The platform helps users identify trends, high-risk areas, and make better decisions. It saves time, reduces technical barriers, and broadens access to valuable insights, contributing to safer roads.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 Hardware Requirements

- Processor: A modern multi-core processor such as Intel Core i5 or equivalent, ensuring smooth performance during data analysis and visualization.
- RAM: At least 8 GB of RAM to handle dataset processing and interactive features effectively.
- Storage: Minimum 256 GB of storage space to store application files, datasets, and user-generated reports.
- Network: Stable internet connectivity with at least 5 Mbps speed to support dynamic data fetching and real-time visualization updates.
- Display: A display resolution of at least 1366x768 pixels for optimal rendering of charts, graphs, and user interface components.

2.2 Software Requirements

- Operating System: Compatible with Windows 10 or later, macOS, or popular Linux distributions to ensure broad accessibility.
- IDE/Text Editor: Visual Studio Code or any preferred code editor to develop and manage the project's codebase efficiently.
- Version Control: Git for source code management and collaboration, with GitHub or GitLab as the remote repository.
- Browser: Modern web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari to run and test the web application smoothly.

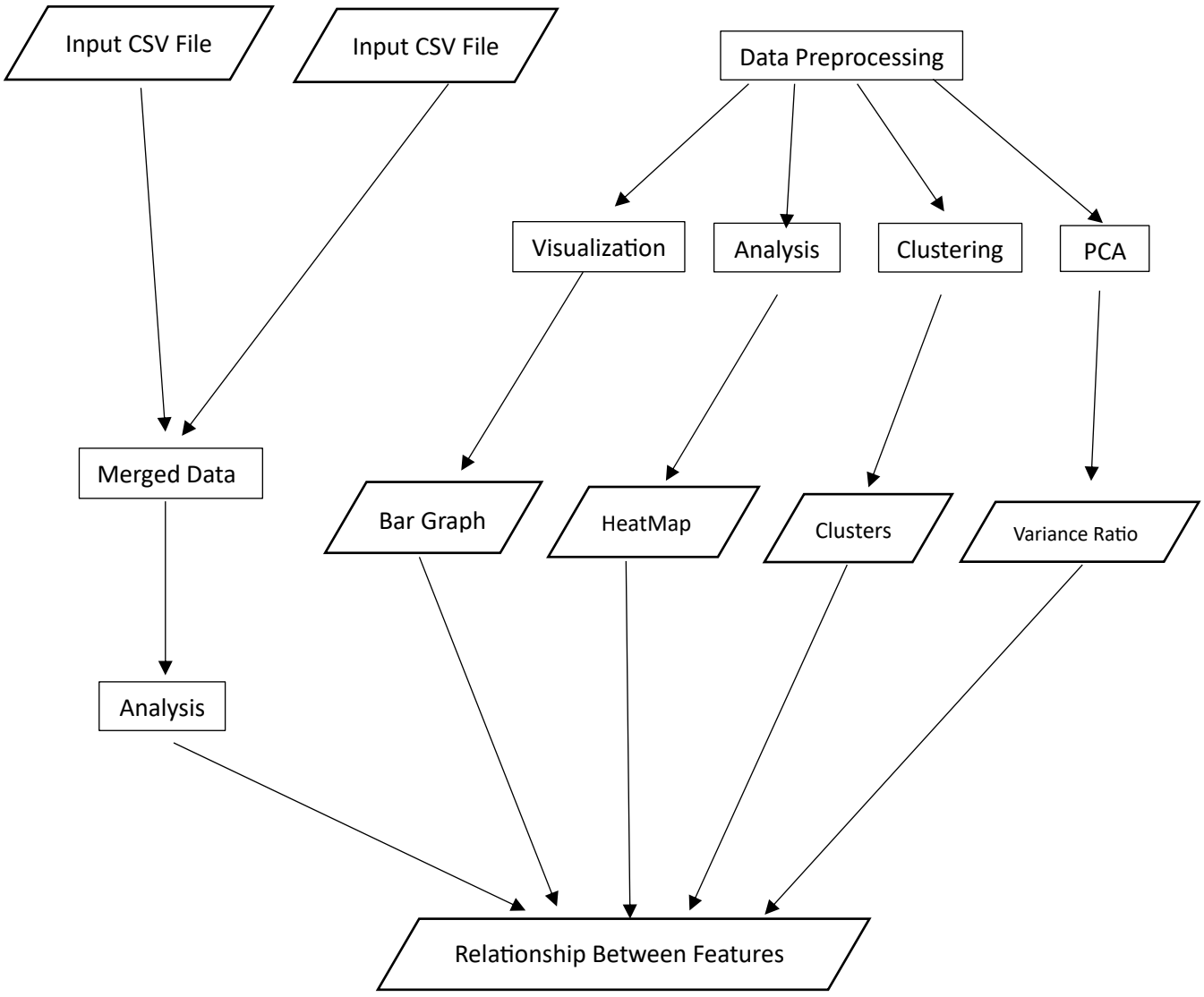
2.3 Technologies Used

- Python: Python is the primary programming language used for data preprocessing and backend development. Its extensive libraries and ease of use make it ideal for handling large datasets and performing complex data analysis.
- Streamlit: Streamlit is a Python-based framework used to build the interactive web application. It simplifies creating dynamic and user-friendly interfaces with minimal coding, allowing quick deployment of data visualization tools.
- Pandas: Pandas is a powerful Python library for data manipulation and analysis. It is used to clean, filter, and transform the traffic accident dataset efficiently before visualization.
- Matplotlib and Seaborn: These libraries provide advanced plotting and visualization capabilities. They are used to create insightful charts and graphs that help users understand trends and patterns in the accident data.
- Git: Git is a version control system that helps manage changes in the project codebase. It supports collaboration and tracks code history, ensuring smooth development workflows.

2.4 Data Flow Diagram (DFD)

- Users interact with the web interface to request data analysis.
- The system processes the data using Python and relevant libraries.
- Processed data is sent to the frontend for visualization.
- Users view dynamic charts, filter results, and export reports.
- The DFD highlights key processes like data retrieval, filtering, and presentation.

1 – level DFD Diagram



CHAPTER 3

Software / Project Design

3.1 Proposed Methodology

The project follows a systematic approach to analyze and visualize traffic accident data. The methodology is divided into phases, each focusing on a critical part of the process to ensure accuracy, meaningful insights, and user interactivity.

3.1.1 Data Collection

- Objective: Obtain reliable and comprehensive traffic accident data.
- Details: The dataset is sourced from FiveThirtyEight, provided in CSV format. This dataset contains state-level information about traffic accidents, including fatal collisions, speeding, alcohol impairment, and other relevant factors.

3.1.2 Data Preprocessing

- Objective: Prepare raw data for analysis by cleaning and transforming it.
- Steps:
  - Handle missing values: Detect and treat missing or null values to maintain dataset integrity. Critical missing data rows are removed, while non-critical gaps may be imputed with suitable values (mean, median, or mode).
  - Convert date/time formats: Ensure all date and time-related data are in standard formats for consistent analysis.
  - Remove duplicates: Eliminate any repeated records to avoid bias.
  - Feature engineering: Extract new meaningful features such as the day of the week or time of day from timestamp columns. These features help uncover temporal patterns in accident occurrences.

Table 3.1.1: Sample Data Set used for Analysis

	State	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
0	Alabama	18.8	39	30	80
1	Alaska	18.1	41	25	94
2	Arizona	18.6	35	28	96
3	Arkansas	22.4	18	26	95
4	California	12.0	35	28	89

3.1.3 Exploratory Data Analysis (EDA)

- Objective: Understand the data distribution and identify trends or anomalies.
- Visualizations:
  - Number of accidents per month/year: Line charts or bar graphs to show temporal trends.
  - Heatmaps: Visual maps depicting accident density geographically or by time.
  - Severity distribution: Pie charts or histograms illustrating the proportion of accidents by severity.
- Outcome: Insights gained from EDA guide subsequent modeling and help identify potential areas of concern or focus.

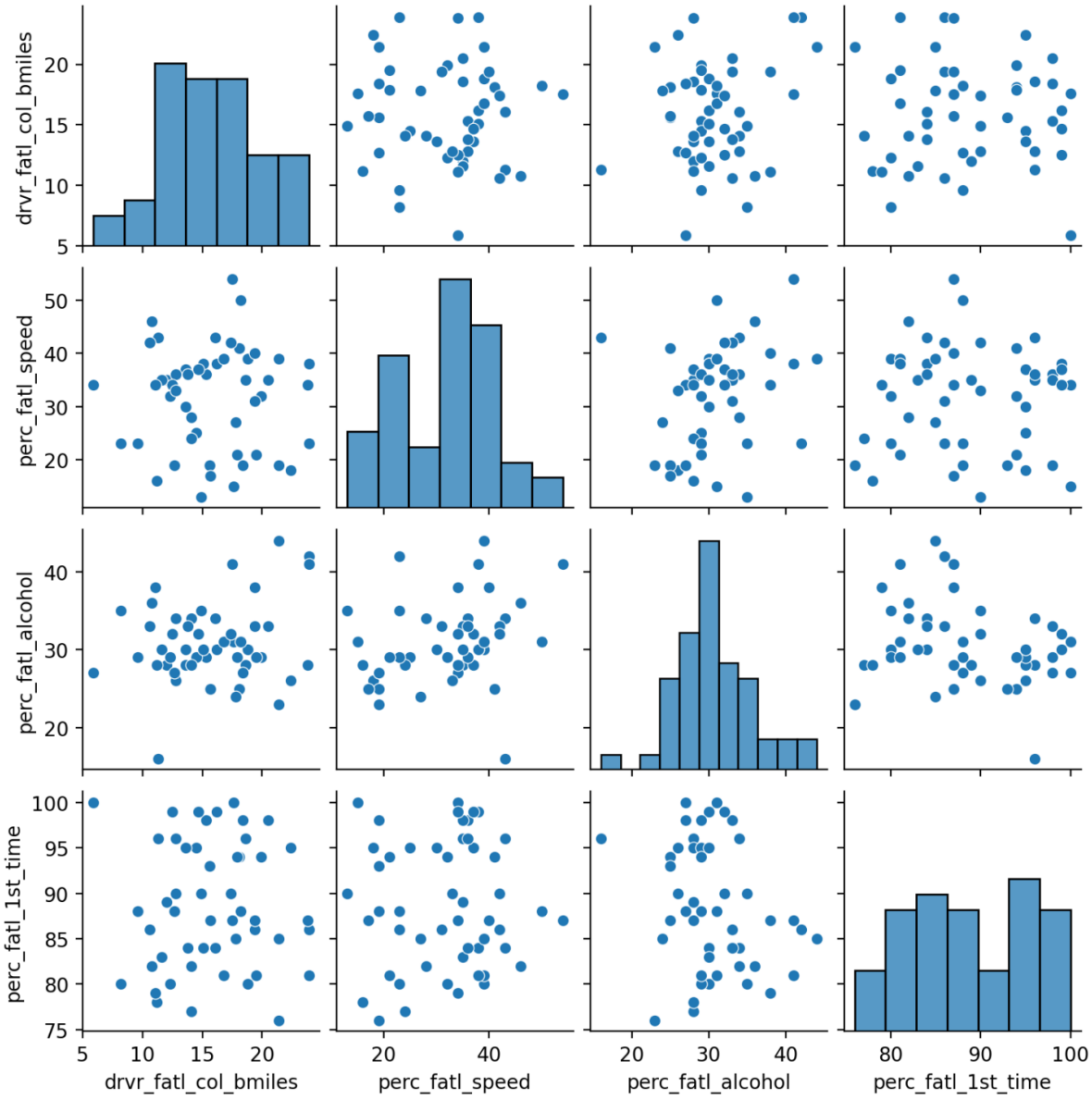


Figure 3.1: Graphical summary of the Data

3.1.4 Dashboard Development (Streamlit)

- Objective: Provide an interactive platform for users to explore data and analysis results.
- Features:
  - Sidebar filters: Allow users to filter data by date range, location (state or city), severity level, or other criteria.
  - Interactive charts: Dynamic visualizations that update based on user input, including scatter plots, bar charts, PCA plots, and cluster visualizations.
  - User-friendly interface: Clear titles, tooltips, and instructions to enhance user experience.

3.2 Algorithm (Example)

This section provides a stepwise approach describing the key algorithmic steps to clean data and produce visualizations in the dashboard.

Algorithm: Data Cleaning and Visualization

1. Load dataset using Pandas.
2. Check for missing values and handle them:
   a. Remove rows with critical missing data.
   b. Impute missing values where appropriate.
3. Parse date-time columns into standard datetime objects.
4. Create new features from date-time:
   - Extract 'Day of Week' from dates.
   - Extract 'Hour of Day' from time.
5. Perform Exploratory Data Analysis:
   - Plot histogram of accidents by hour to detect peak times.
   - Generate heatmap showing accident locations or densities.
6. Develop the Streamlit dashboard:
   - Implement sidebar filters for date, location, and severity.
   - Display charts and graphs dynamically based on filter selections.

Explanation:

- Loading data: Using Pandas, the CSV file is imported for manipulation.
- Cleaning: Handling missing and duplicate data ensures high data quality.
- Feature engineering: Adding temporal features enables time-based insights.
- EDA: Visualizations reveal patterns essential for informed analysis.
- Streamlit app: Provides the interface for interactive exploration, allowing stakeholders to gain insights without programming knowledge.

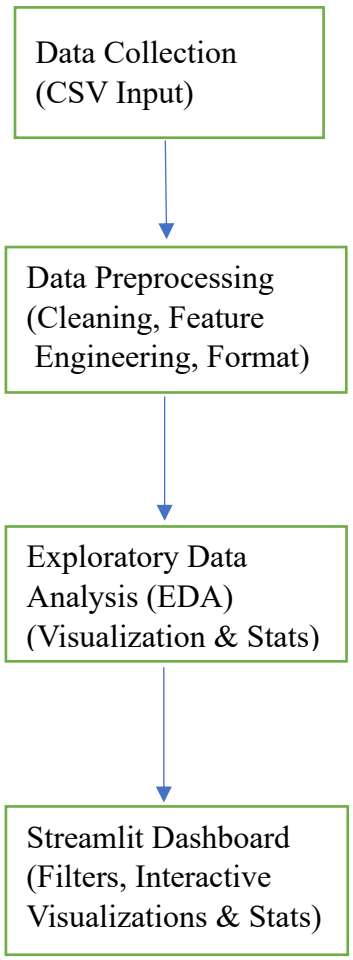
3.3 Blueprint / Architecture Diagram

The architecture diagram visually represents the flow of data and processes within the project. It shows how raw data is transformed and presented through the system.

Textual block diagram:

[Data Source (CSV)] → [Data Cleaning] → [Exploratory Data Analysis (EDA)] → [Dashboard Visualization]

Detailed block diagram:



Explanation:

- Data Collection: The entry point where raw CSV data is imported.
- Data Preprocessing: Cleaning and transforming data for consistency.
- EDA: Generating insights through statistical summaries and visualizations.
- Dashboard: Delivering an interactive user interface for data exploration.

CHAPTER 4

RESULT/TESTING OF PROJECT / SOFTWARE

4.1 Results

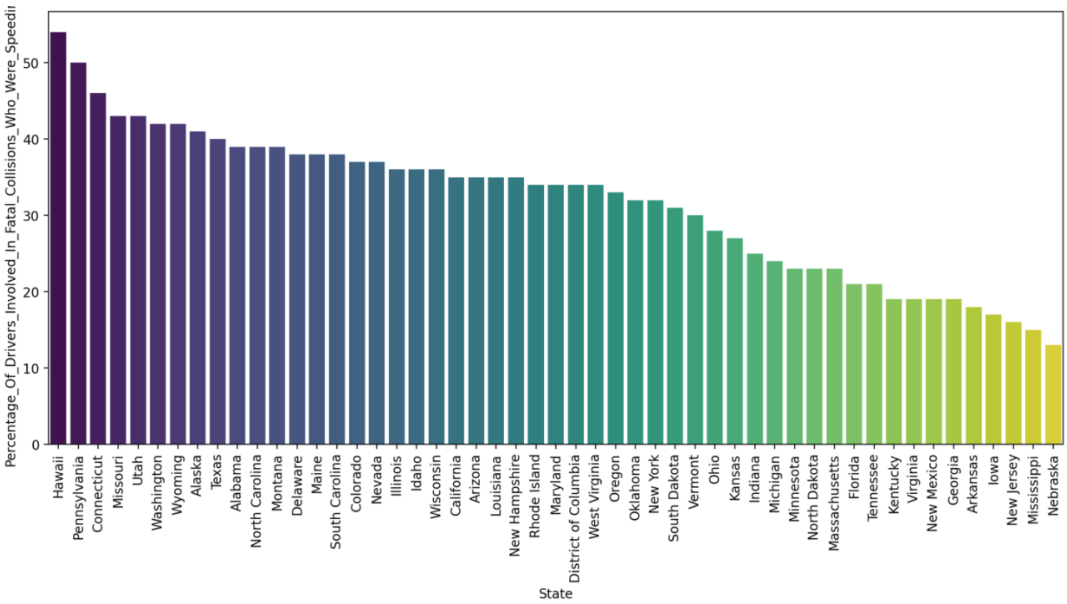


Figure 4.1.1: State-level Comparative Analysis

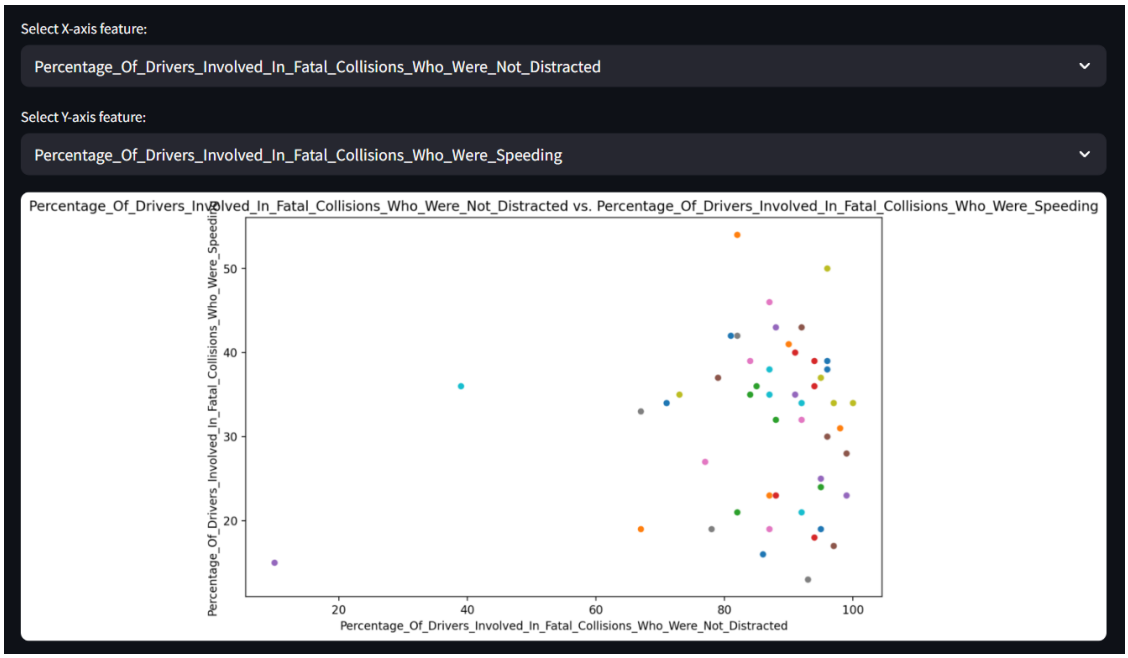
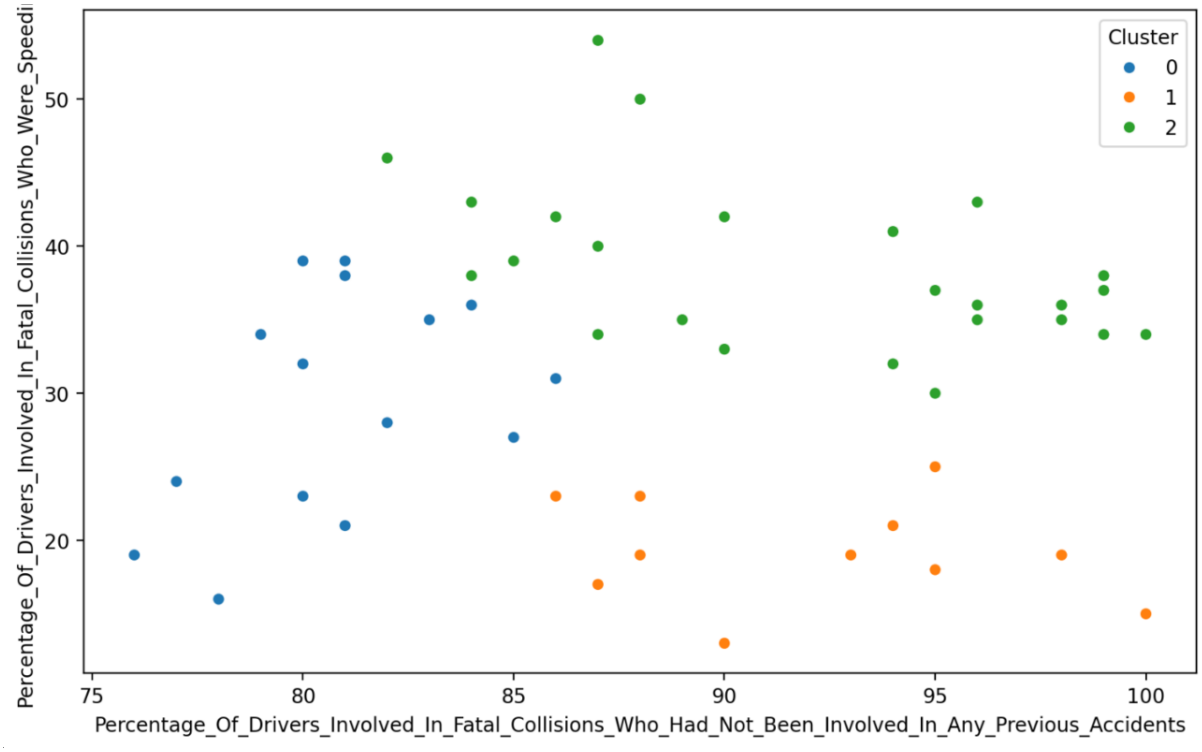


Figure 4.1.2: Correlation and Relationship Analysis



. Figure 4.1.3: Cluster Analysis (K-Means)

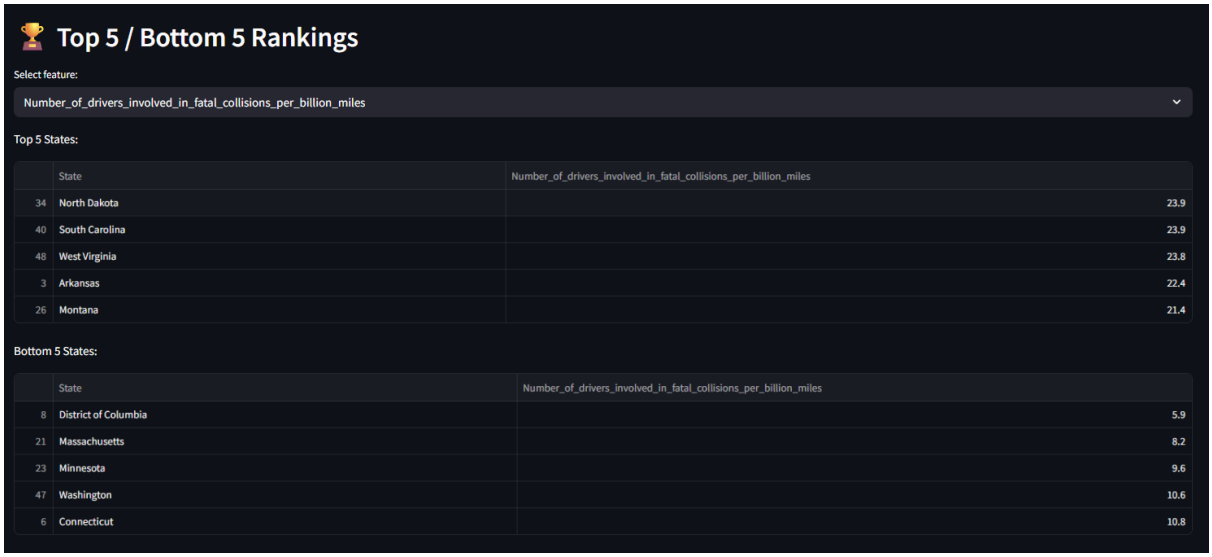


Figure 4.1.4. Top-Bottom Ranking of States Based on Feature selected

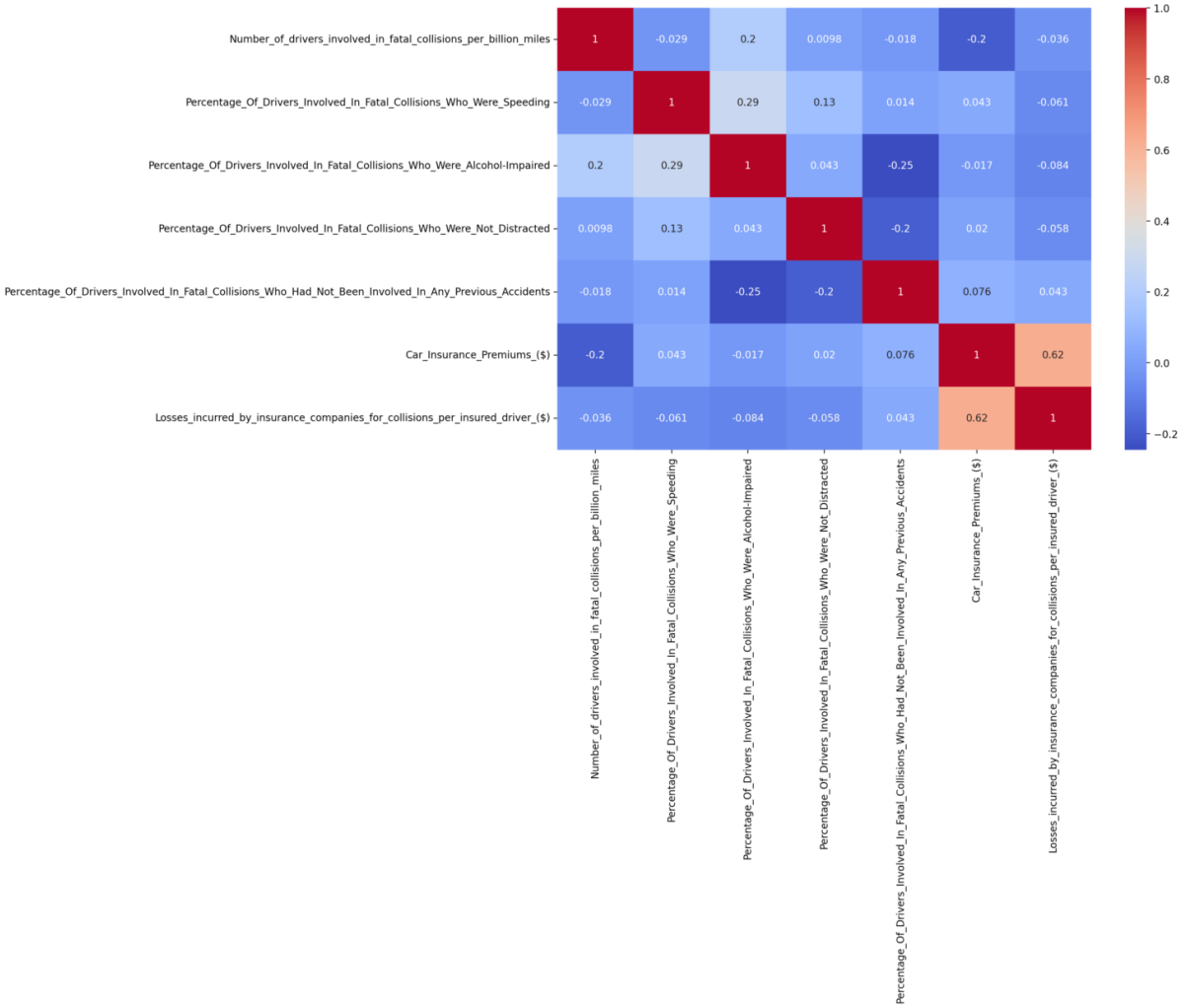


Figure 4.1.5: Correlation Heatmap

4.2 Testing

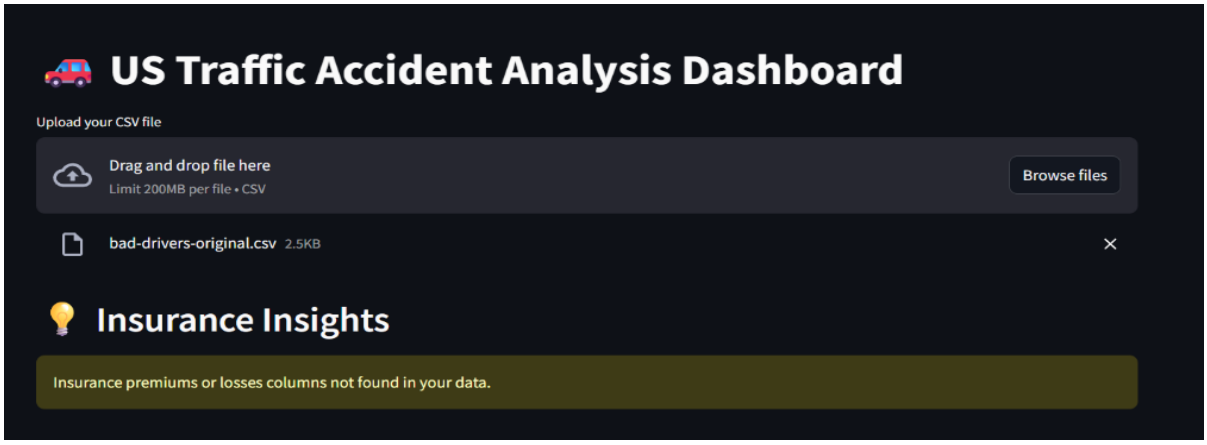


Figure 4.2.1: Data cleaning functions: Test with incomplete data.

Figure 4.2.2: Visualizations: Validate correct rendering

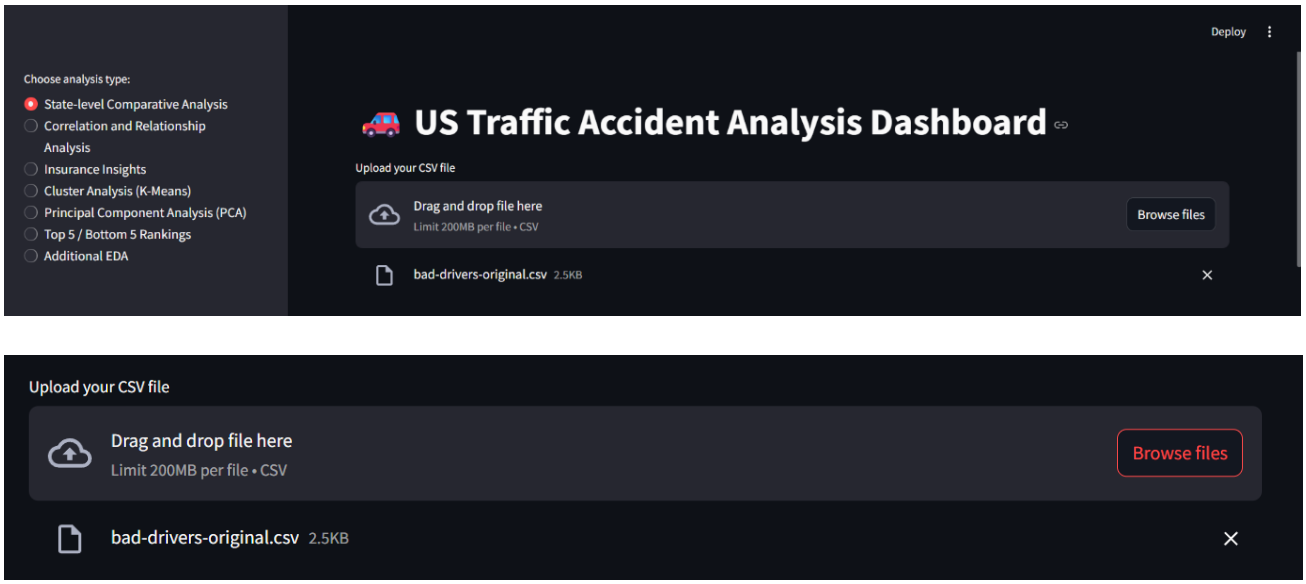


Figure 4.2.5: Test the dashboard's usability (buttons, filters).

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1. Conclusion

This project highlights the successful development of an interactive, user-friendly dashboard that enables comprehensive analysis of US traffic accident data at the state level. By integrating various statistical and machine learning techniques—such as correlation analysis, clustering, and principal component analysis—the platform provides valuable insights into patterns and factors influencing traffic accidents. Users can explore relationships between different metrics, identify high-risk states, and understand insurance-related trends, which can support policymakers, safety organizations, and insurance companies in making data-driven decisions. Overall, this project demonstrates the power of combining data visualization and analytical methods to transform complex datasets into actionable knowledge that can ultimately contribute to improving road safety and reducing traffic-related losses.

5.2 Future Scope

- **Incorporate Real-Time Data:** Integrate live traffic and accident data feeds to provide up-to-date insights and timely alerts.
- **Expand Geographic Coverage:** Include accident data from other countries or regions to enable comparative international analysis.
- **Add More Data Sources:** Combine with weather, road condition, and vehicle data to improve the understanding of accident causes.
- **Predictive Modeling:** Develop advanced machine learning models to predict accident risks based on historical trends and external factors.
- **User Personalization:** Allow users to customize dashboards and reports based on their specific interests or roles (e.g., policymakers, insurance agents).
- **Mobile Application:** Create a mobile-friendly version or app for easier access and usage on the go.
- **Integration with Safety Campaigns:** Use insights to support argeted awareness programs and road safety interventions.
- **Enhanced Visualization:** Incorporate interactive maps, heatmaps, and time series animations for deeper exploratory analysis.
- **Driver Behavior Analysis:** Analyze driver behavior patterns from telematics data to correlate with accident likelihood.
- **Insurance Risk Assessment:** Develop tools for insurance companies to better assess risk and personalize premiums.
- **Collaboration Platform:** Enable sharing of insights and data among government agencies, researchers, and public safety organizations.
- **Automated Reporting:** Generate automated summaries and recommendations based on the data analysis for decision-makers.

APPENDIX (CODE)

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Page Configuration
st.set_page_config(page_title="US Traffic Accident Analysis", layout="wide")

# Title
st.title("🚗 US Traffic Accident Analysis Dashboard")

# File Upload
uploaded_file = st.file_uploader("Upload your CSV file", type=["csv"])

if uploaded_file is not None:
    df = pd.read_csv(uploaded_file)

    # Rename columns for easier access
    df.columns = df.columns.str.strip().str.replace(' ', '_').str.replace('%', 'Pct')

    # Sidebar Navigation
    analysis_type = st.sidebar.radio(
        "Choose analysis type:",
        [
            "State-level Comparative Analysis",
            "Correlation and Relationship Analysis",
            "Insurance Insights",
            "Cluster Analysis (K-Means)",
            "Principal Component Analysis (PCA)",
            "Top 5 / Bottom 5 Rankings",
            "Additional EDA"
        ]
    )

    # 1 State-level Comparative Analysis
    if analysis_type == "State-level Comparative Analysis":
        st.header("📊 State-level Comparative Analysis")
        selected_metric = st.selectbox("Select metric:", df.columns[1:])
        ranked_df = df[["State", selected_metric]].sort_values(by=selected_metric,
ascending=False)
        st.dataframe(ranked_df)
        fig, ax = plt.subplots(figsize=(14, 6))
```

```
        sns.barplot(x='State', y=selected_metric, data=ranked_df, palette='viridis')
        plt.xticks(rotation=90)
        st.pyplot(fig)

    # 2 Correlation and Relationship Analysis
    elif analysis_type == "Correlation and Relationship Analysis":
        st.header("🔍 Correlation and Relationship Analysis")
        x_feature = st.selectbox("Select X-axis feature:", df.columns[1:])
        y_feature = st.selectbox("Select Y-axis feature:", df.columns[1:])
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.scatterplot(x=x_feature, y=y_feature, data=df, hue='State', palette='tab10',
legend=False)
        plt.xlabel(x_feature)
        plt.ylabel(y_feature)
        plt.title(f"{x_feature} vs. {y_feature}")
        st.pyplot(fig)

        st.subheader("Correlation Heatmap")
        corr = df.select_dtypes(include='number').corr()
        fig, ax = plt.subplots(figsize=(12, 8))
        sns.heatmap(corr, annot=True, cmap='coolwarm')
        st.pyplot(fig)

    # 3 Insurance Insights
    elif analysis_type == "Insurance Insights":
        st.header("💡 Insurance Insights")
        if 'Insurance_premiums' in df.columns and 'Losses_incurred' in df.columns:
            fig, ax = plt.subplots(figsize=(14, 6))
            sns.boxplot(x='State', y='Insurance_premiums', data=df, palette='Set2')
            plt.xticks(rotation=90)
            plt.title("Insurance Premiums by State")
            st.pyplot(fig)

            fig, ax = plt.subplots(figsize=(14, 6))
            sns.barplot(x='State', y='Losses_incurred', data=df, palette='Set3')
            plt.xticks(rotation=90)
            plt.title("Losses Incurred by State")
            st.pyplot(fig)

            corr_val = df['Insurance_premiums'].corr(df['Losses_incurred'])
            st.write(f"Correlation between insurance premiums and losses: **{corr_val:.2f}**")
        else:
            st.warning("Insurance premiums or losses columns not found in your data.")

    # 4 Cluster Analysis (K-Means)
    elif analysis_type == "Cluster Analysis (K-Means)":
```

```
        st.header("🔗 Cluster Analysis (K-Means)")
        features = st.multiselect("Select features for clustering:",
df.select_dtypes(include='number').columns.tolist())
        if len(features) >= 2:
            X = df[features].dropna()
            k = st.slider("Number of clusters:", 2, 10, 3)
            kmeans = KMeans(n_clusters=k, random_state=42).fit(X)
            df['Cluster'] = kmeans.labels_
            fig, ax = plt.subplots(figsize=(10, 6))
            sns.scatterplot(x=features[0], y=features[1], hue='Cluster', data=df, palette='tab10')
            st.pyplot(fig)
            st.dataframe(df[['State'] + features + ['Cluster']])
        else:
            st.warning("Please select at least two features.")

    # 5 Principal Component Analysis (PCA)
    elif analysis_type == "Principal Component Analysis (PCA)":
        st.header("🧩 Principal Component Analysis (PCA)")
        features = st.multiselect("Select features for PCA:",
df.select_dtypes(include='number').columns.tolist())
        if len(features) >= 2:
            X = df[features].dropna()
            pca = PCA(n_components=2)
            components = pca.fit_transform(X)
            df['PCA1'] = components[:, 0]
            df['PCA2'] = components[:, 1]
            fig, ax = plt.subplots(figsize=(10, 6))
            sns.scatterplot(x='PCA1', y='PCA2', hue='State', data=df, palette='tab20')
            plt.xlabel('PCA Component 1')
            plt.ylabel('PCA Component 2')
            st.pyplot(fig)
            st.write("Explained Variance Ratio:", pca.explained_variance_ratio_)
        else:
            st.warning("Please select at least two features.")

    # 6 Top 5 / Bottom 5 Rankings
    elif analysis_type == "Top 5 / Bottom 5 Rankings":
        st.header("🏆 Top 5 / Bottom 5 Rankings")
        selected_feature = st.selectbox("Select feature:", df.columns[1:])
        top5 = df.sort_values(by=selected_feature, ascending=False).head(5)
        bottom5 = df.sort_values(by=selected_feature, ascending=True).head(5)
        st.write("Top 5 States:")
        st.dataframe(top5[['State', selected_feature]])
        st.write("Bottom 5 States:")
        st.dataframe(bottom5[['State', selected_feature]])
```

```
    # 7 Additional EDA
    elif analysis_type == "Additional EDA":
        st.header("📈 Additional Exploratory Data Analysis")

        st.subheader("Boxplot")
        selected_box_feature = st.selectbox("Select feature for boxplot:",
df.select_dtypes(include='number').columns.tolist())
        fig, ax = plt.subplots(figsize=(14, 6))
        sns.boxplot(x='State', y=selected_box_feature, data=df, palette='Set1')
        plt.xticks(rotation=90)
        st.pyplot(fig)

        st.subheader("Histogram")
        selected_hist_feature = st.selectbox("Select feature for histogram:",
df.select_dtypes(include='number').columns.tolist())
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.histplot(df[selected_hist_feature], bins=20, kde=True, color='skyblue')
        plt.title(f"Distribution of {selected_hist_feature}")
        st.pyplot(fig)

        st.subheader("Summary Statistics")
        st.dataframe(df.describe())

        st.subheader("Correlation Heatmap")
        corr = df.select_dtypes(include='number').corr()
        fig, ax = plt.subplots(figsize=(12, 8))
        sns.heatmap(corr, annot=True, cmap='coolwarm')
        st.pyplot(fig)

else:
    st.info("Please upload a CSV file to start the analysis.")
```

REFERENCES

Dataset Source

- FiveThirtyEight, "Bad Drivers Data," https://github.com/fivethirtyeight/data/tree/master/bad-drivers

Python and Libraries

- Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56.
- Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95.
- Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021.
- Streamlit Documentation, https://docs.streamlit.io

Machine Learning Algorithms

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R. Springer.
  - Covers linear regression, KMeans clustering, and principal component analysis (PCA).
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
  - A deeper dive into clustering, dimensionality reduction, and regression techniques.
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
  - General reference for supervised and unsupervised learning algorithms.

Additional Online Resources

- Towards Data Science, “Traffic Accident Data Analysis with Python,” https://towardsdatascience.com
- Kaggle Notebooks and Datasets, https://www.kaggle.com

Government and Regulatory Data

- National Highway Traffic Safety Administration (NHTSA), https://www.nhtsa.gov