JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD TELANGANA - 500085

"A ROAD ACCIDENT PREDICTION MODEL USING DATA MINING TECHNIQUES"

Project report submitted in partial fulfilment of the
Requirements for the award of the Degree of
Bachelor of Technology
in
CSE (Data Science)
By

SHIVANI AGGIDI	**(21P71A6783)**
N SRI CHARAN	**(21P71A6794)**
KAHKASHA UROOJ	**(21P71A67A4)**
BHADRAKSH CHAPALLI	**(21P71A67C1)**

Under the Esteemed Guidance of
Mrs. T. SHAILAJA , Assistant Professor
Department of CSE (Data Science)

SWAMI VIVEKANANDA INSTITUTE OF TECHNOLOGY
Under the Management of Mahbub College (MHSS)
Mahbub College Campus, Patny Centre, S.D. Road, Secunderabad - 500003, T.S.

SWAMI VIVEKANANDA INSTITUTE OF TECHNOLOGY
Mahbub College Campus, R.P Road, Secunderabad-500003
(Affiliated to JNTUH)
2021-2025

SWAMI VIVEKANANDA INSTITUTE OF TECHNOLOGY
Under the Management of Mahbub College (MHSS)
Mahbub College Campus, Patny Centre, S.D. Road, Secunderabad - 500003, T.S.

CERTIFICATE

This is to certify that the project report entitled "**A ROAD ACCIDENT PREDICTION MODEL USING DATA MINING TECHNIQUES**" is submitted by Shivani Aggidi (21P71A6783), N Sri Charan (21P71A6794), Kahkasha Urooj (21P71A67A4), Bhadraksh Chapalli (21P71A67C1) in partial fulfilment for the award of Degree of BACHELOR OF TECHNOLOGY in CSE (DATA SCIENCE), to the Jawaharlal Nehru Technological University-Hyderabad is a record of Bonafide work carried out by them under my guidance and supervision.

Date:

Internal Guide

HOD – CSE (DS)

External Examiner Principal

ACKNOWLEDGEMENT

First, we thank our Project Guide Associate Professor Mrs. T. SHAILAJA of the Department of CSE (Data Science) for giving us an opportunity for developing this project. This project has really helped us in enhancing our skills of programming, the perspective with which we should view projects and above all, our presentation, and interpersonal skills. Last but not the least, we also thank the professors of our department for their help and wishes for successful completion of project.

Shivani Aggidi

21P71A6783

N. Sri Charan

21P71A6794

Kahkasha Urooj

21P71A67A4

Bhadraksh Chapalli

21P71A67C1

DECLARATION

We hereby declare that the work which is being presented in this Major Project entitled, **"A ROAD ACCIDENT PREDICTION MODEL USING DATA MINING TECHNIQUES"** submitted to JNTU-H, in the partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in CSE (DATA SCIENCE), is an authentic record of our own work carried out from January 2025 to May 2025 under the supervision of Mrs. M Supriya Samuel, Associate Professor and HOD, Department of CSE (Data Science), SVIT, Mahbub Campus.

The matter embodied in this project report has not been submitted by me for the award of any other degree.

Place: SECUNDERABAD

Date:

Shivani Aggidi

21P71A6783

N. Sri Charan

21P71A6794

Kahkasha Urooj

21P71A67A4

Bhadraksh Chapalli

21P71A67C1

INDEX

ABSTRACT

Due to the exponentially increasing number of vehicles on the road, the number of accidents occurring on a daily basis is also increasing at an alarming rate. With the high number of traffic incidents and deaths these days, the ability to forecast the number of traffic accidents over a given time is important for the transportation department to make scientific decisions. In this scenario, it will be good to analyse the occurrence of accidents so that this can be further used to help us in coming up with techniques to reduce them. Even though uncertainty is a characteristic trait of majority of the accidents, over a period of time, there is a level of regularity that is perceived on observing the accidents occurring in a particular area. This regularity can be made use of in making well informed predictions on accident occurrences in an area and developing accident prediction models. In this paper, we have studied the inter relationships between road accidents, condition of a road and the role of environmental factors in the occurrence of an accident. We have made use of data mining techniques in developing an accident prediction model using Apriori algorithm and Support Vector Machines. Bangalore road accident datasets for the years 2014 to 2017 available in the internet have been made use for this study. The results from this study can be advantageously used by several stakeholders including and not limited to the government public work departments, contractors and other automobile industries in better designing roads and vehicles based on the estimates

LIST OF FIGURES

1. INTRODUCTION

The alarming rate of increase of accidents in India is now a cause for serious concern. According to some recent statistics, India accounts for roughly six percent of global road accidents while owning only one percent of the global vehicle population. There are a lot of accident cases reported due to the negligence of two-wheelers, whereas over-speeding is also another contributing factor. Accidents caused while under the influence of alcohol or during general traffic violations are also common. In spite of having set regulations and the highway codes, the negligence of people towards the speed of the vehicle, the vehicle condition and their own negligence of not wearing helmets has caused a lot of accidents. While the major cause of road accidents is attributed to the increasing number of vehicles, the role played by the condition of the roads and other environmental factors cannot be overlooked.

The number of deaths due to road accidents in India is indeed a cause for worry. The scenario is very dismal with more than 137,000 people succumbing to injuries from road accidents. This figure is more than four times the annual death toll from terrorism. Accidents involving heavy goods vehicles like trucks and even those involving commercial vehicles used for public transportation like buses are some of the most fatal kinds of accidents that occur, claiming the lives of innocent people. Weather conditions like rain, fog, etc., also play a role in catalysing the risk of accidents. Thus, having a proper estimation of accidents and knowledge of accident hotspots and ca sing factors will help in taking steps to reduce them. This requires a keen study on accidents and development of accident prediction models.

2. LITERATURE SURVEY

BeshahT.,&Hill,S.(2010)

Developed a decision tree model to analyse traffic accident data and predict accident severity.
The study emphasized how rule-based classification techniques can assist in understanding accident patterns and identifying high-risk situations.

Chang,L.Y.,&Chen,W.C.(2005)

Applied Bayesian network models to predict accident involvement based on driver behaviour and environmental factors.
The results showed improved prediction accuracy by integrating multiple risk variables.

Sharma,A.,&MukherjeeS.(2015)

Used random forest algorithms on traffic datasets to predict accident hotspots.
Their approach proved effective in handling large datasets with high dimensionality and gave better insights into accident-prone zones.

Khan,A.M.,&Razzak,M.I.(2019)

Proposed an ensemble learning approach combining decision trees and SVMs for road accident prediction.
The model achieved higher accuracy and reduced false positives compared to individual algorithms.

Sohn,S.Y.,&Lee,S.H.(2003)

Utilized data mining to identify key accident factors such as road surface, weather, and lighting conditions.
Their study concluded that environmental factors significantly influence accident occurrence.

Ramos,F.M.,&Almeida,P.S.(2018)

Applied clustering techniques to categorize accident data and detect patterns in urban vs. rural areas. Their model helped authorities deploy preventive measures based on accident density and severity clusters.

3. SYSTEM REQUIREMENTS AND ANALYSIS

3.1. EXISTING SYSTEM

The existing systems for road accident prediction primarily rely on traditional statistical methods such as logistic regression and basic historical data analysis to identify accident-prone areas and factors. These systems often focus on individual parameters like traffic volume, weather conditions, or road types without integrating multiple variables simultaneously. Moreover, many current models are static in nature, lacking the ability to adapt to real-time changes or learn from newly generated data. These conventional approaches also depend heavily on manual data interpretation, which limits their scalability and accuracy in complex, dynamic traffic environments. While some systems utilize basic machine learning algorithms, they often suffer from data imbalance issues, poor handling of noisy or missing data, and limited predictive performance due to lack of feature engineering and model optimization. As a result, the existing systems struggle to provide accurate, actionable insights for proactive accident prevention and traffic management

DISADVANTAGES

- Limited Data Integration:
- Low Accuracy in Complex Scenarios:
- Lack of Real-Time Prediction
- Poor Handling of Missing or Noisy Data
- Minimal Visualization and User Interface

3.2. PROPOSED SYSTEM

The proposed system aims to enhance the accuracy and efficiency of road accident prediction by leveraging advanced data mining techniques and machine learning algorithms. Unlike traditional models, this system integrates a wide range of data sources including

historical accident records, traffic volume, weather conditions, road types, vehicle categories, and driver behaviours. It employs classification algorithms such as Random Forest, Decision Trees, and Support Vector Machines (SVM) to analyse patterns and predict accident probabilities in specific locations or scenarios. Additionally, clustering methods are used to identify accident hotspots and categorize risk zones based on severity and frequency. The model is designed to be adaptive, enabling continuous learning from new data to improve prediction performance over time. Furthermore, the system supports real-time data integration, allowing dynamic analysis and timely alerts to authorities and drivers. Through visual dashboards and intuitive interfaces, traffic management systems can make informed decisions and implement preventive measures effectively. The proposed solution is scalable, data-driven, and capable of providing actionable insights for safer road environments.

ADVANTAGES

- Improved Prediction Accuracy
- Multi-Factor Analysis:
- Adaptive Learning
- Hotspot Identification

3.3. HARDWARE/SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS:

- System : Intel Core I3
- Hard Disk : 256 GB.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 4 Gb.

SOFTWARE REQUIREMENTS:

- Operating System: Windows
- Coding Language: Python 3.7

3.4 SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system**.**

4. SYSTEM DESIGN

4.1. SYSTEM ARCHITECTURE

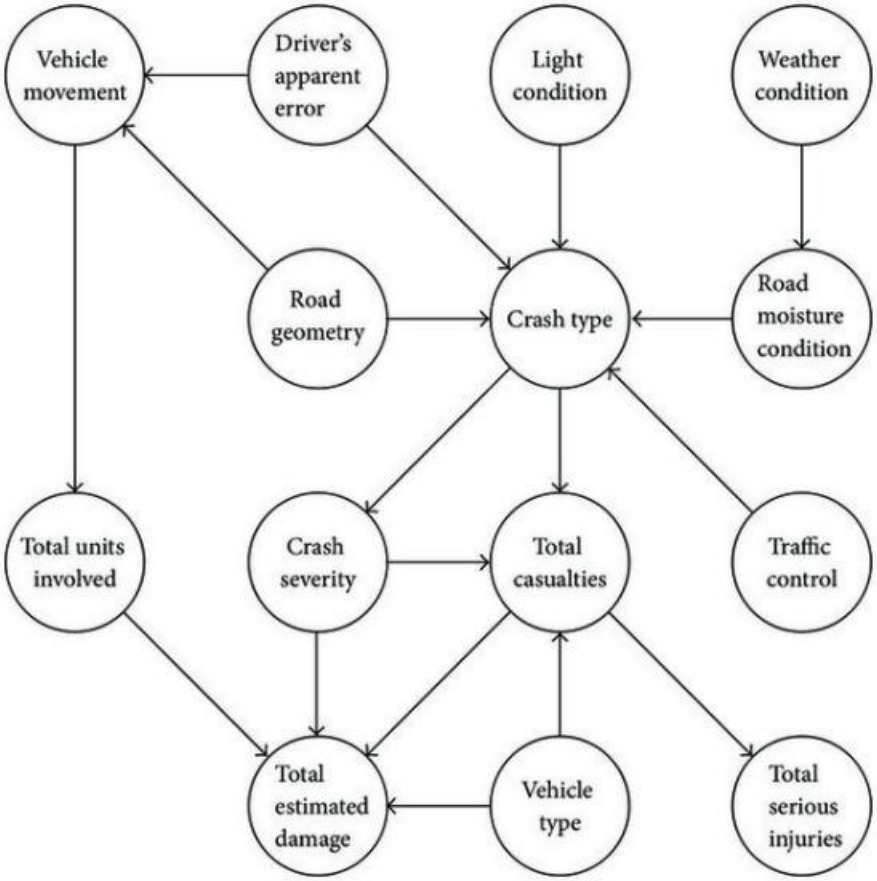


Fig. 4.1.System Architecture

4.2. UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

4.2.1.USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

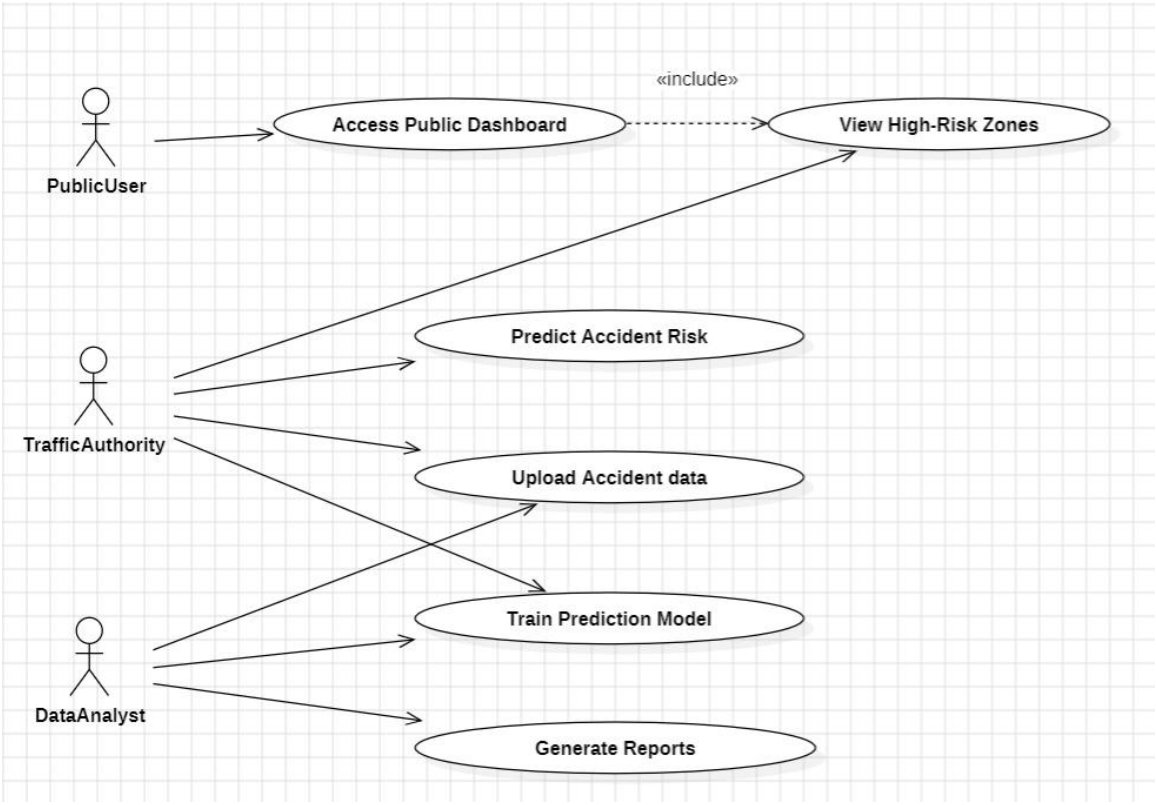


Fig. 4.2.Use case Diagram

4.2.2.CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

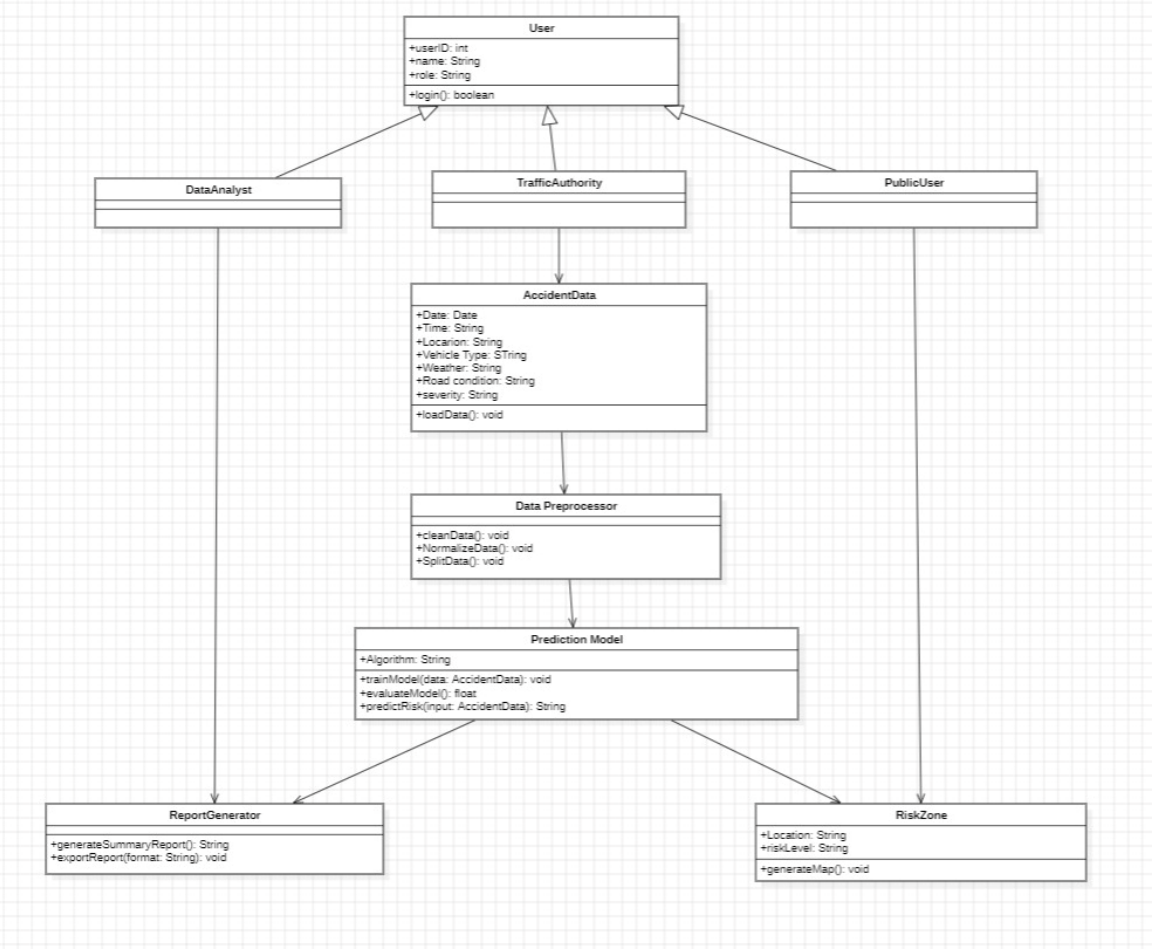


Fig. 4.3.Class Diagram

4.2.3.SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

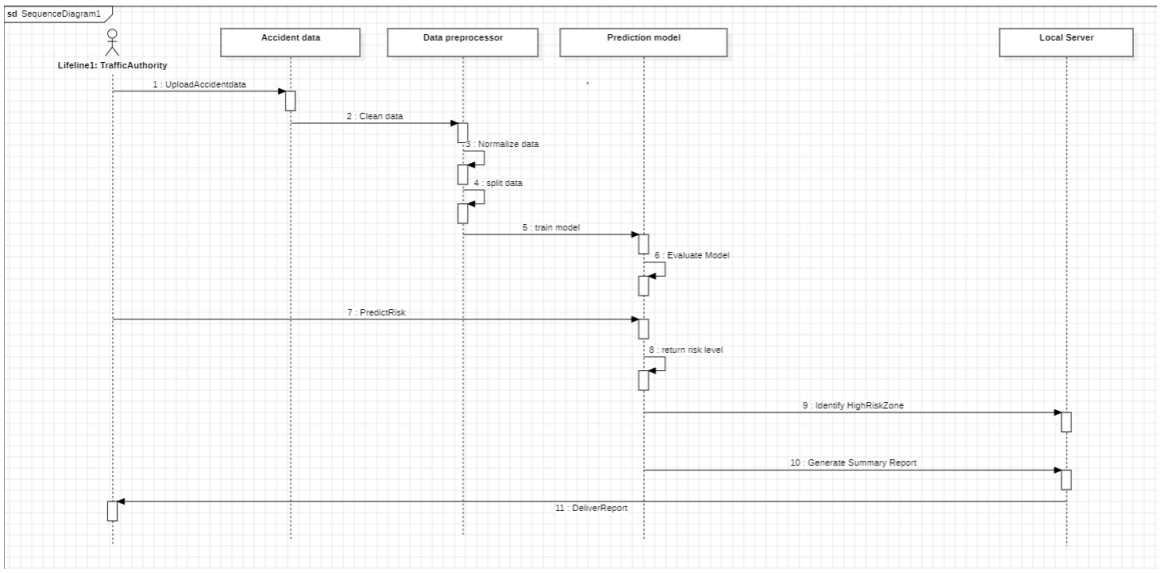


Fig. 4.4.Sequence Diagram

4.2.4.ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

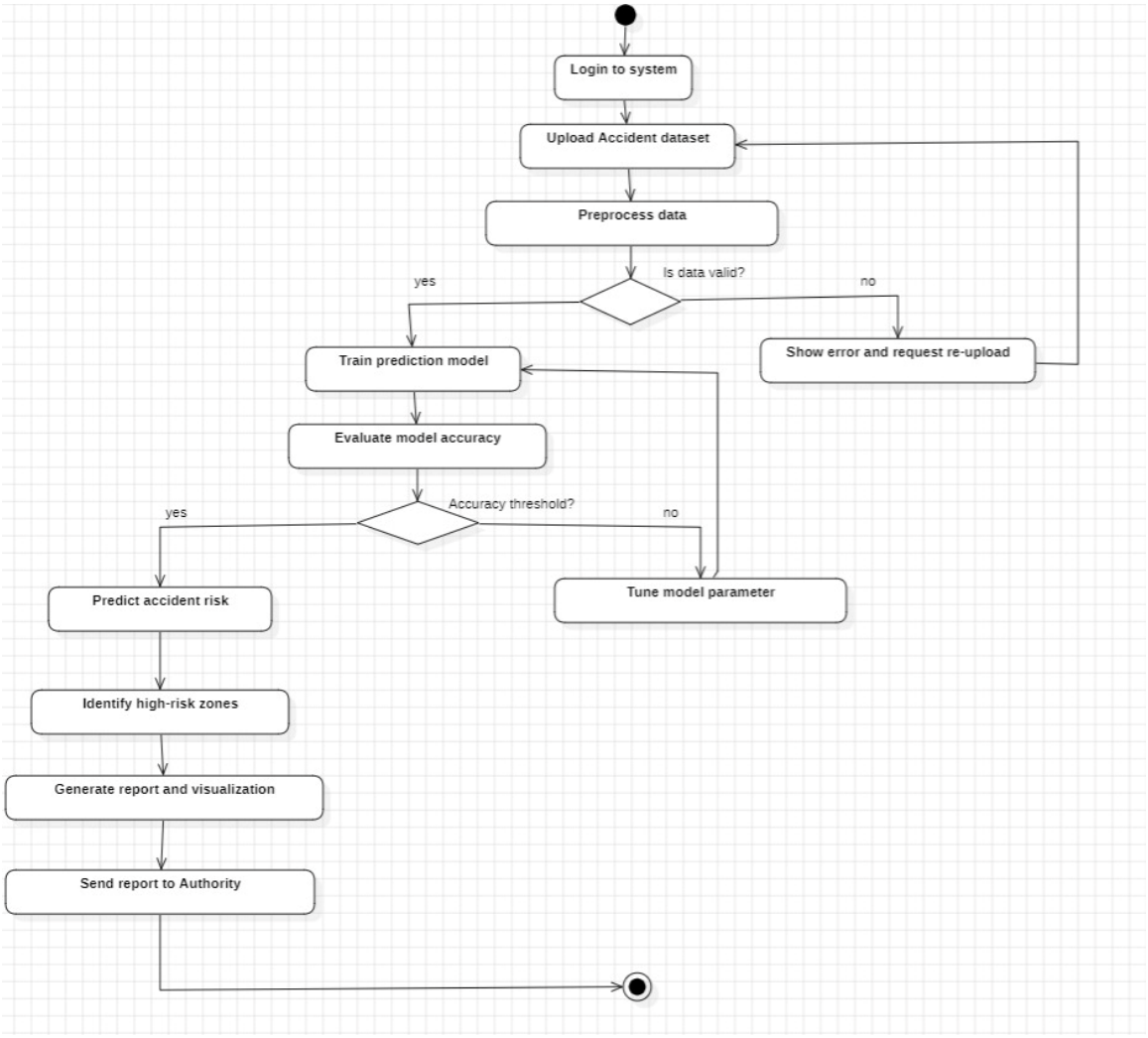


Fig. 4.5.Activity Diagram

5. INPUT AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

6. IMPLEMENTATION

Tech Stack:

Language: Python

Libraries: Pandas, Scikit-learn, Matplotlib/Seaborn, Flask (optional for UI/API)

Model: Logistic Regression / Random Forest / Decision Tree

Dataset: Road accident dataset (can be from Kaggle or government open data)

7. SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQtetc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python:-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.**

This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** These further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you download for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point

for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java Database Connectivity)** and **ODBC (Open Database Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

History of Python: -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence

the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voorWiskundeen Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. " Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning: -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tenable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools

effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Learning:-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modelling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be

used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning:-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning:-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine

Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So, it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like colour, smell, taste, etc.

- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labelled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviours that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users

to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close

to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. " Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

TensorFlow

TensorFlow is a free and source software across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code

- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyse. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and I Python shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**

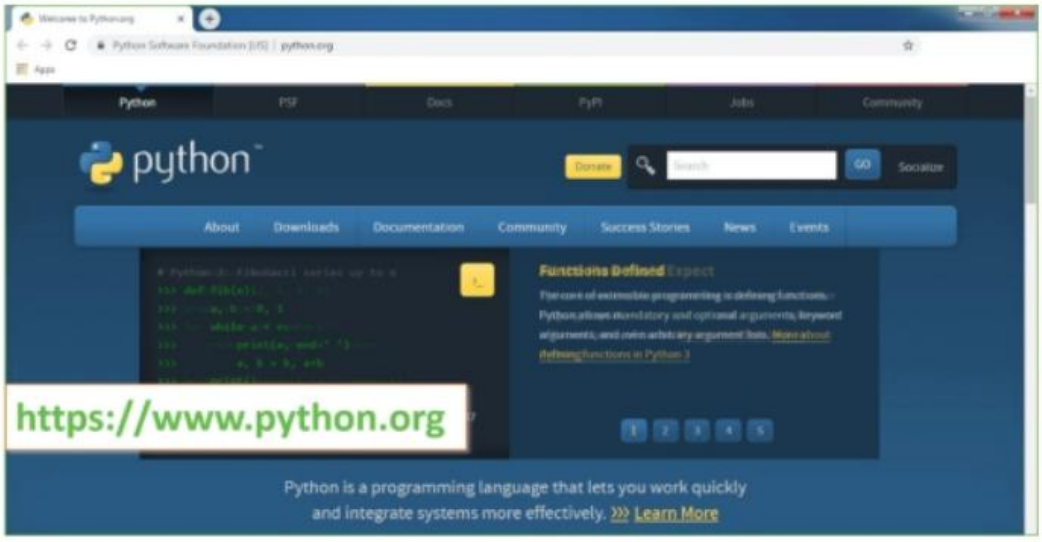


Fig..7.1.Download Python

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Fig. 7.2.Latest Version

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Colour or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

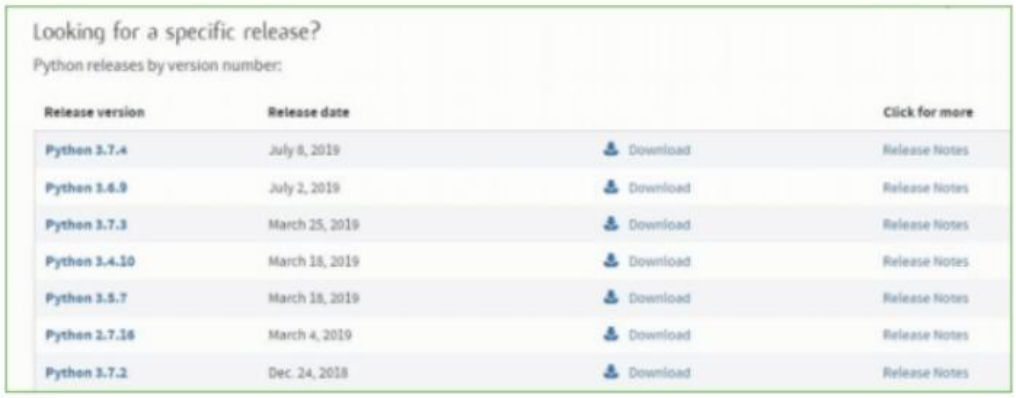


Fig. 7.3.Downloading Latest version

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

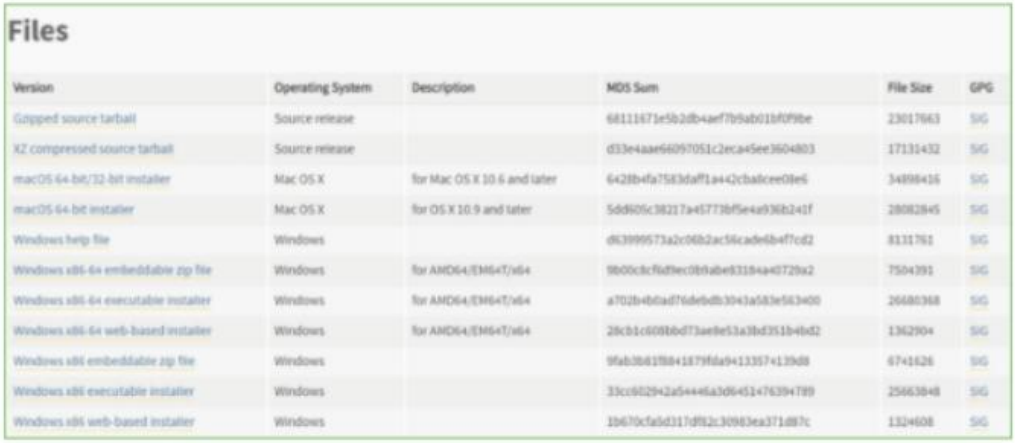


Fig. 7.4.Verifying files

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

• To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

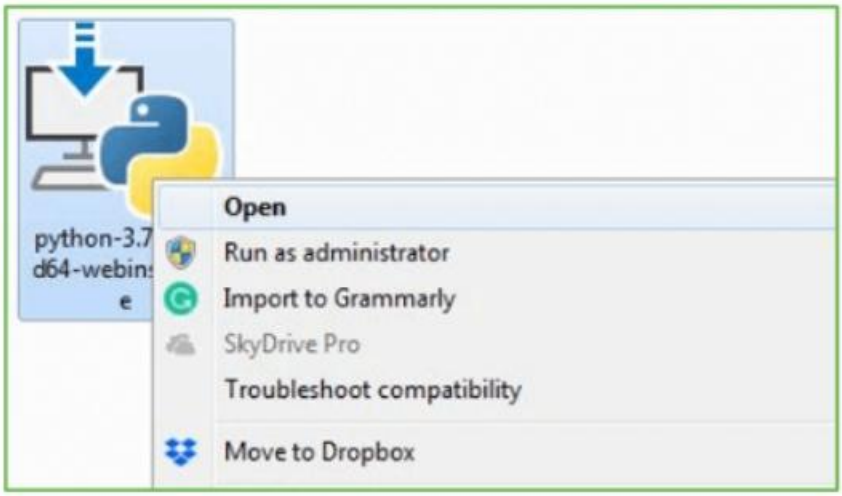


Fig. 7.5.Open Python

Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.

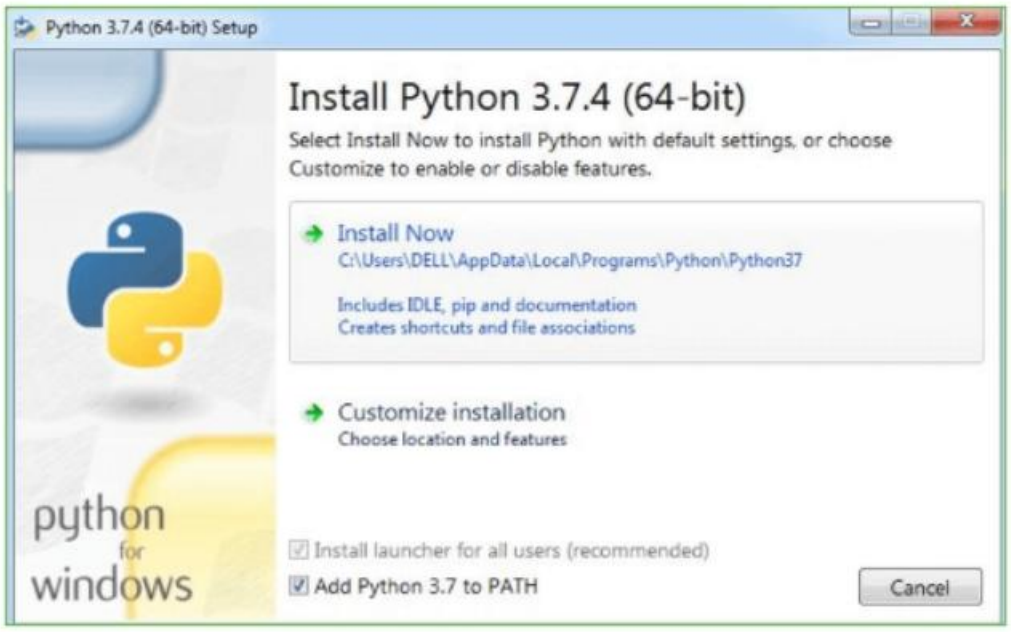


Fig. 7.6.Install Python

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Fig. 7.7.Setup Successful

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

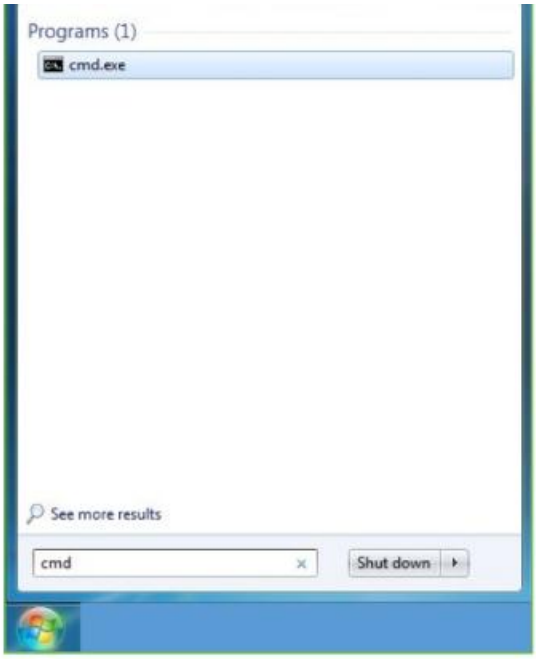


Fig7.8.Open CMD

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.

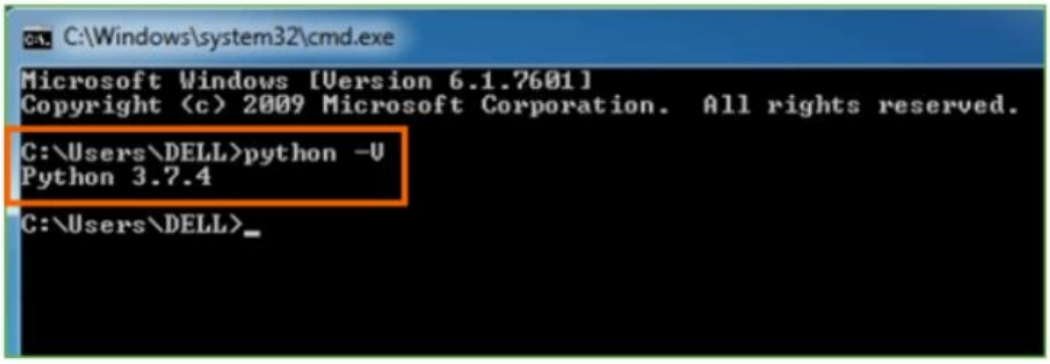


Fig. 7.9.Check Python Version

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".

Fig. 7.10.Run python idle

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**

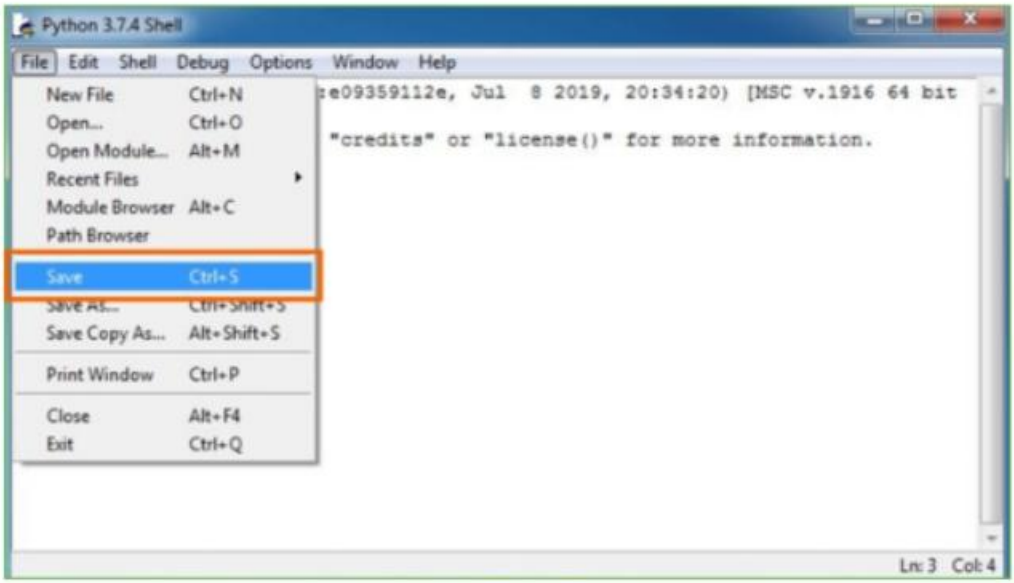


Fig. 7.11.Save File

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World. Now for e.g. **enter print**

8. CODE SNIPPETS

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
"""Run administrative tasks."""
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'a_road_accident_prediction.settings')
try:
from django.core.management import execute_from_command_line
except ImportError as exc:
raise ImportError(
"Couldn't import Django. Are you sure it's installed and "
"available on your PYTHONPATH environment variable? Did you "
"forget to activate a virtual environment?"
) fromexc
execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':

main()
```

SETTINGS.PY

```
import os


# Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/


# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = 'm+1edl5m-5@u9u!b8-=4-4mq&o1%agco2xpl8c!7sn7!eowjk#'


# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True


ALLOWED_HOSTS = []


# Application definition


INSTALLED_APPS = [
```

```
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'Remote_User',
'Service_Provider',
]

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'a_road_accident_prediction.urls'
```

```
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [(os.path.join(BASE_DIR,'Template/htmls'))],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]

WSGI_APPLICATION = 'a_road_accident_prediction.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
```

```
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'a_road_accident_prediction',
'USER':'root',
'PASSWORD': '',
'HOST' :'127.0.0.1',
'PORT' :'3306',
}
}

# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
{
'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
```

```
},
{
'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]

# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/
```

```
STATIC_URL = '/static/'

STATICFILES_DIRS = [os.path.join(BASE_DIR,'Template/images')]

MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'Template/media')


STATIC_ROOT = '/static/'


STATIC_URL = '/static/'
```

Views.py remote user

```
from django.db.models import Count

from django.db.models import Q

from django.shortcuts import render, redirect, get_object_or_404

import datetime

import openpyxl


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

from sklearn.ensemble import VotingClassifier
```

```
import warnings
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
# Create your views here.
from Remote_User.models import ClientRegister_Model,road_accident_prediction,detection_ratio,detection_accuracy


def login(request):


if request.method == "POST" and 'submit1' in request.POST:


username = request.POST.get('username')
password = request.POST.get('password')
try:
enter = ClientRegister_Model.objects.get(username=username,password=password)
request.session["userid"] = enter.id
```

```
return redirect('ViewYourProfile')
except:
pass

return render(request,'RUser/login.html')

def Register1(request):

if request.method == "POST":
username = request.POST.get('username')
email = request.POST.get('email')
password = request.POST.get('password')
phoneno = request.POST.get('phoneno')
country = request.POST.get('country')
state = request.POST.get('state')
city = request.POST.get('city')
ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
country=country, state=state, city=city)

return render(request, 'RUser/Register1.html')
else:
```

```
return render(request,'RUser/Register1.html')


def ViewYourProfile(request):
userid = request.session['userid']
obj = ClientRegister_Model.objects.get(id= userid)
return render(request,'RUser/ViewYourProfile.html',{'object':obj})


def Predict_Road_Accident_Status(request):
expense = 0
kg_price=0
if request.method == "POST":


Reference_Number= request.POST.get('Reference_Number')
State= request.POST.get('State')
Area_Name= request.POST.get('Area_Name')
Traffic_Rules_Viaolation= request.POST.get('Traffic_Rules_Viaolation')
Vechile_Load= request.POST.get('Vechile_Load')
Time= request.POST.get('Time')
Road_Class= request.POST.get('Road_Class')
Road_Surface= request.POST.get('Road_Surface')
Lighting_Conditions= request.POST.get('Lighting_Conditions')
Weather_Conditions= request.POST.get('Weather_Conditions')
```

```
Person_Type= request.POST.get('Person_Type')

Sex= request.POST.get('Sex')

Age= request.POST.get('Age')

Type_of_Vehicle= request.POST.get('Type_of_Vehicle')


df = pd.read_csv('Road_Accidents.csv')

df

df.columns

df.rename(columns={'Label': 'label', 'Reference_Number': 'RId'}, inplace=True)


def apply_results(label):

if (label == 0):

return "No Accident"

elif (label == 1):

return "Accident"


df['results'] = df['label'].apply(apply_results)

results = df['results'].value_counts()

# df.drop(['Road Surface','LightingConditions','Sex','Age','label','Type of Vehicle','Person Type'],axis=1,inplace=True)


cv = CountVectorizer(lowercase=False)
```

```
y = df['results']

# X = df.drop("results", axis=1)

X = df["RId"].apply(str)


print("X Values")

print(X)

print("Labels")

print(y)


X = cv.fit_transform(X)


models = []

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

X_train.shape, X_test.shape, y_train.shape


# SVM Model

print("SVM")

from sklearn import svm

lin_clf = svm.LinearSVC()

lin_clf.fit(X_train, y_train)
```

```
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))

print("KNeighborsClassifier")
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier()
kn.fit(X_train, y_train)
knpredict = kn.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, knpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, knpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, knpredict))
models.append(('KNeighborsClassifier', kn))
```

```
classifier = VotingClassifier(models)

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)


rno = [Reference_Number]

vector1 = cv.transform(rno).toarray()

predict_text = lin_clf.predict(vector1)


pred = str(predict_text).replace("[", "")

pred1 = pred.replace("]", "")

prediction = re.sub("[^a-zA-Z]", " ", str(pred1))


road_accident_prediction.objects.create(Reference_Number=Reference_Number,

State=State,

Area_Name =Area_Name,

Traffic_Rules_Viaolation=Traffic_Rules_Viaolation,

Vechile_Load=Vechile_Load,

Time=Time,

Road_Class=Road_Class,

Road_Surface=Road_Surface,

Lighting_Conditions=Lighting_Conditions,

Weather_Conditions=Weather_Conditions,
```

Person_Type=Person_Type,

Sex=Sex,

Age=Age,

Type_of_Vehicle=Type_of_Vehicle,

SVM=prediction)

return render(request, 'RUser/Predict_Road_Accident_Status.html',{'objs':prediction})

return render(request, 'RUser/Predict_Road_Accident_Status.html')

views.py service provider

from django.db.modelsimport Count, Avg

from django.shortcuts import render, redirect

from django.db.models import Count

from django.db.models import Q

import datetime

import xlwt

from django.http import HttpResponse

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

```
from sklearn.ensemble import VotingClassifier
import warnings
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import openpyxl


# Create your views here.
from Remote_User.models import ClientRegister_Model,road_accident_prediction,detection_ratio,detection_accuracy



def serviceproviderlogin(request):
if request.method  == "POST":
admin = request.POST.get('username')
password = request.POST.get('password')
if admin == "Admin" and password =="Admin":
return redirect('View_Remote_Users')
```

```
return render(request,'SProvider/serviceproviderlogin.html')


def viewtreandingquestions(request,chart_type):

dd = {}

pos,neu,neg =0,0,0

poss=None

topic = road_accident_prediction.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')

for t in topic:

topics=t['ratings']

pos_count=road_accident_prediction.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))

poss=pos_count

for pp in pos_count:

senti= pp['names']

if senti == 'positive':

pos= pp['topiccount']

elifsenti == 'negative':

neg = pp['topiccount']

elifsenti == 'nutral':

neu = pp['topiccount']
```

```
dd[topics]=[pos,neg,neu]

return
render(request,'SProvider/viewtreandingquestions.html',{'object':topic,'dd':dd,'chart_type':c
hart_type})


def View_All_Road_Accident_Prediction(request):


obj = road_accident_prediction.objects.all()

return render(request, 'SProvider/View_All_Road_Accident_Prediction.html', {'objs': obj})


def Find_Road_Accident_Prediction_Type_Ratio(request):

detection_ratio.objects.all().delete()

ratio = ""

kword = ' No Accident'

print(kword)

obj = road_accident_prediction.objects.all().filter(Q(SVM=kword))

obj1 = road_accident_prediction.objects.all()

count = obj.count();

count1 = obj1.count();

ratio = (count / count1) * 100

if ratio != 0:

detection_ratio.objects.create(names=kword, ratio=ratio)
```

```
ratio1 = ""

kword1 = ' Accident'

print(kword1)

obj1 = road_accident_prediction.objects.all().filter(Q(SVM=kword1))

obj11 = road_accident_prediction.objects.all()

count1 = obj1.count();

count11 = obj11.count();

ratio1 = (count1 / count11) * 100

if ratio1 != 0:

detection_ratio.objects.create(names=kword1, ratio=ratio1)


obj = detection_ratio.objects.all()

return render(request, 'SProvider/Find_Road_Accident_Prediction_Type_Ratio.html', {'objs': obj})



def View_Remote_Users(request):

obj=ClientRegister_Model.objects.all()

return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})


def ViewTrendings(request):
```

```
topic = road_accident_prediction.objects.values('topics').annotate(dcount=Count('topics')).order_by('-dcount')

return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})


def negativechart(request,chart_type):

dd = {}

pos, neu, neg = 0, 0, 0

poss = None

topic = road_accident_prediction.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')

for t in topic:

topics = t['ratings']

pos_count = road_accident_prediction.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))

poss = pos_count

for pp in pos_count:

senti = pp['names']

if senti == 'positive':

pos = pp['topiccount']

elifsenti == 'negative':
```

```
neg = pp['topiccount']
elifsenti == 'nutral':
neu = pp['topiccount']
dd[topics] = [pos, neg, neu]
return
render(request,'SProvider/negativechart.html',{'object':topic,'dd':dd,'chart_type':chart_type}
)

def charts(request,chart_type):
chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def charts1(request,chart_type):
chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def likeschart(request,like_chart):
charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})
def likeschart1(request,like_chart):
charts =detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
return render(request,"SProvider/likeschart1.html", {'form':charts, 'like_chart':like_chart})
```

```
def Download_Trained_DataSets(request):


response = HttpResponse(content_type='application/ms-excel')
# decide file name
response['Content-Disposition'] = 'attachment; filename="TrainedData.xls"'
# creating workbook
wb = xlwt.Workbook(encoding='utf-8')
# adding sheet
ws = wb.add_sheet("sheet1")
# Sheet header, first row
row_num = 0
font_style = xlwt.XFStyle()
# headers are bold
font_style.font.bold = True
# writer = csv.writer(response)
obj = road_accident_prediction.objects.all()
data = obj  # dummy method to fetch data.
for my_row in data:
row_num = row_num + 1
ws.write(row_num, 0, my_row.Reference_Number, font_style)
ws.write(row_num, 1, my_row.State, font_style)
```

```
ws.write(row_num, 2, my_row.Area_Name, font_style)

ws.write(row_num, 3, my_row.Traffic_Rules_Viaolation, font_style)

ws.write(row_num, 4, my_row.Vechile_Load, font_style)

ws.write(row_num, 5, my_row.Time, font_style)

ws.write(row_num, 6, my_row.Road_Class, font_style)

ws.write(row_num, 7, my_row.Road_Surface, font_style)

ws.write(row_num, 8, my_row.Lighting_Conditions, font_style)

ws.write(row_num, 9, my_row.Weather_Conditions, font_style)

ws.write(row_num, 10, my_row.Person_Type, font_style)

ws.write(row_num, 11, my_row.Sex, font_style)

ws.write(row_num, 12, my_row.Age, font_style)

ws.write(row_num, 13, my_row.Type_of_Vehicle, font_style)

ws.write(row_num, 14, my_row.SVM, font_style)


wb.save(response)

return response


def Train_Test_DataSets(request):


detection_accuracy.objects.all().delete()


df = pd.read_csv('Road_Accidents.csv')
```

```
df

df.columns

df.rename(columns={'Label': 'label'}, inplace=True)

df['Refno']=df['Reference_Number']


def apply_results(label):

if (label == 0):

return "No Accident"

elif (label == 1):

return "Accident"


df['results'] = df['label'].apply(apply_results)

results = df['results'].value_counts()

# df.drop(['Road Surface','LightingConditions','Sex','Age','label','Type of Vehicle','Person Type'],axis=1,inplace=True)


cv = CountVectorizer(lowercase=False)


y = df['results']

# X = df.drop("results", axis=1)

X = df["Refno"].apply(str)
```

```
print("X Values")
print(X)
print("Labels")
print(y)

X = cv.fit_transform(X)

models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
X_train.shape, X_test.shape, y_train.shape
print("X_test")
print(X_test)
print(X_train)

# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
```

```
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))
detection_accuracy.objects.create(names="SVM", ratio=svm_acc)


print("KNeighborsClassifier")
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier()
kn.fit(X_train, y_train)
knpredict = kn.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, knpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, knpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, knpredict))
models.append(('KNeighborsClassifier', kn))
detection_accuracy.objects.create(names="KNeighborsClassifier",
ratio=accuracy_score(y_test, knpredict) * 100)
```

9. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of

components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10. OUTPUT SCREENS

Step1: H**ere** we see the output of successfully run the Django framework and we got link.

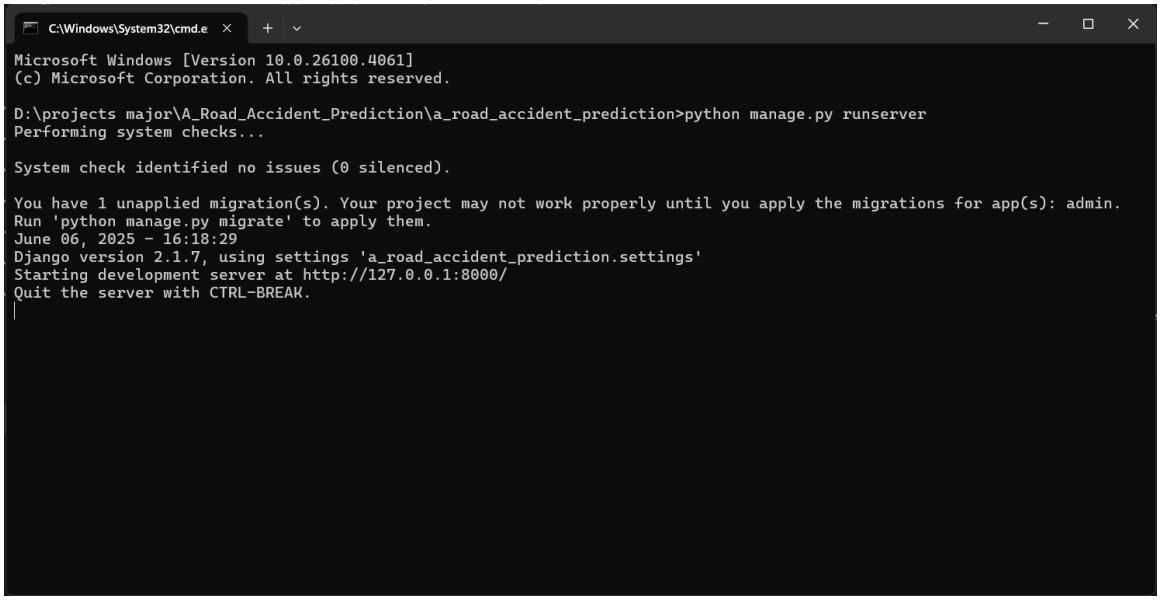


Fig 10.1. Run Django Framework

Step2: Here we see our project output after the link paste the web browser.

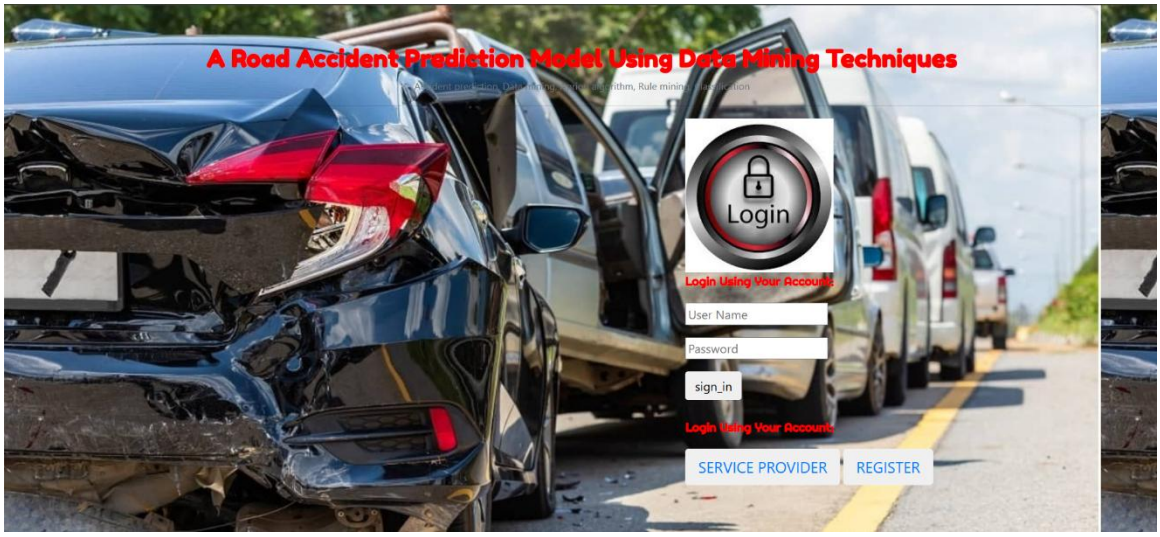


Fig. 10.2. Project interface

Step3: You are for new use we have to register

Fig. 10.3.Register

Step4: After login this will be displayed

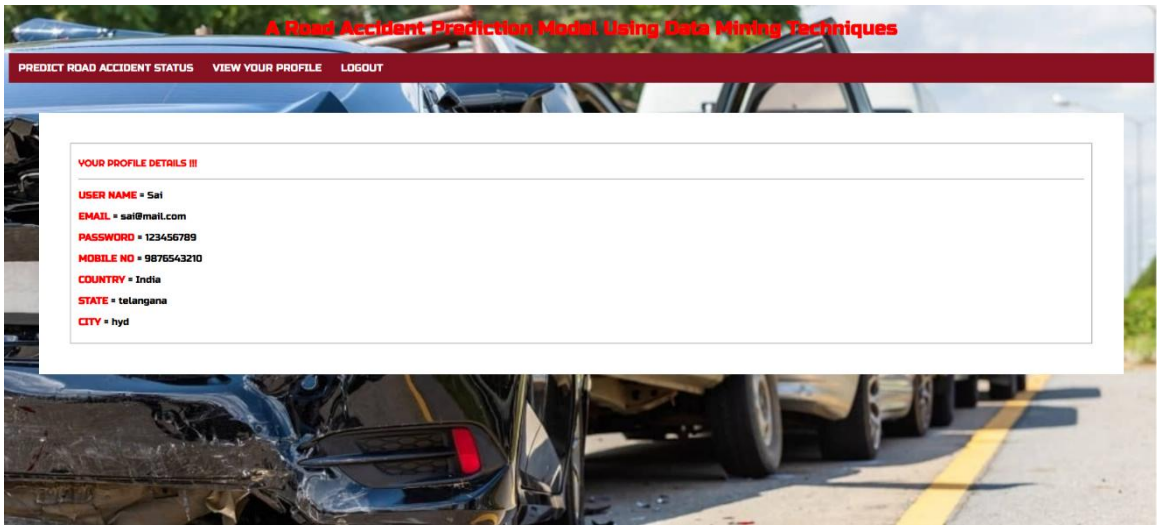


Fig. 10.4.User profile Display

Step5: We predict the road accidents we have data we have to predict it.

Above results will be displayed below and logout.

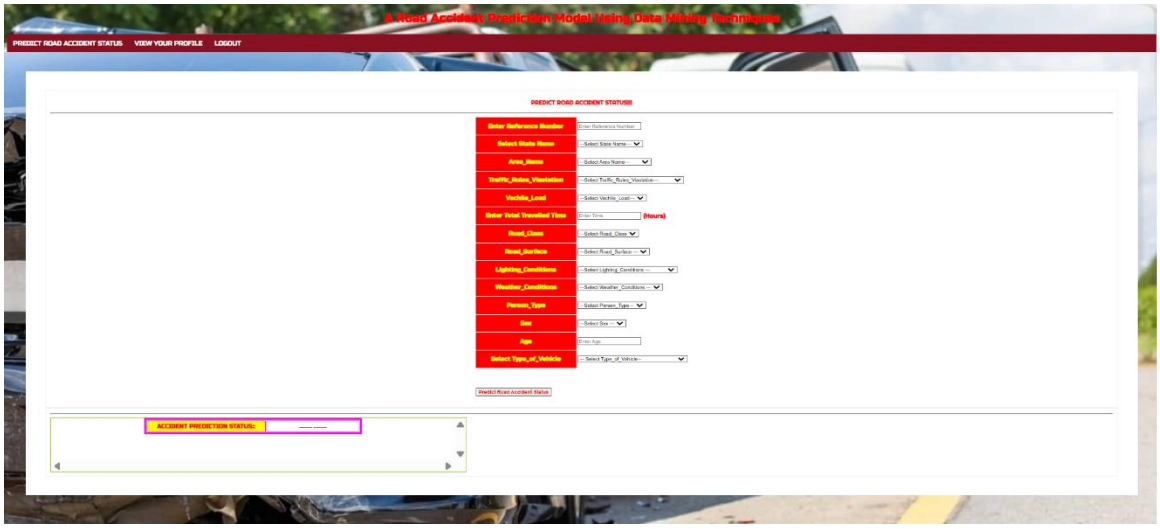


Fig. 10.5. Predicting road accident

Step6: After logging the service provider logging above screen will be displayed.

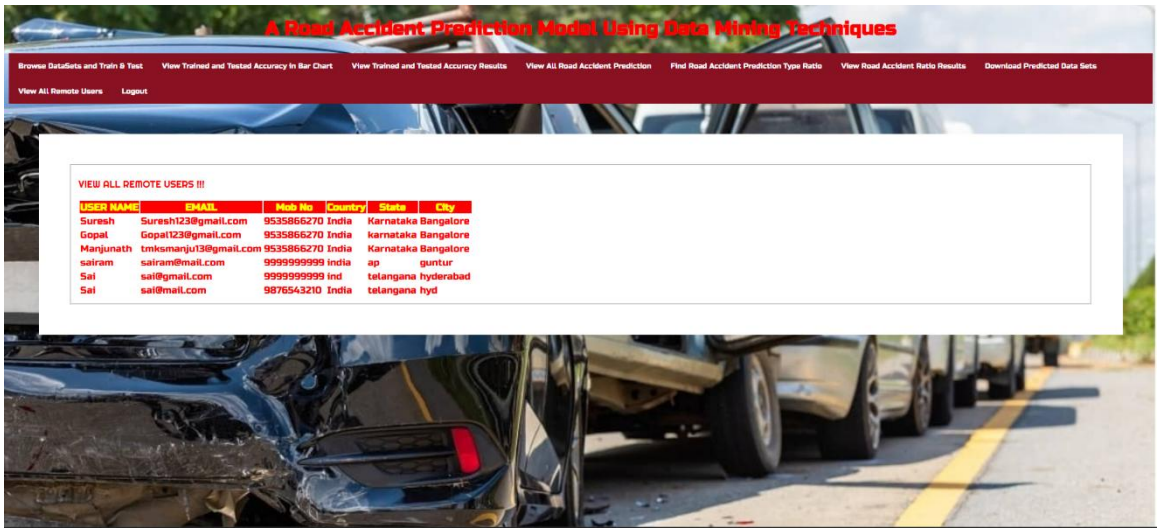


Fig. 10.6.Remote users display

Step7: Here we see the algorithms and its accuracy

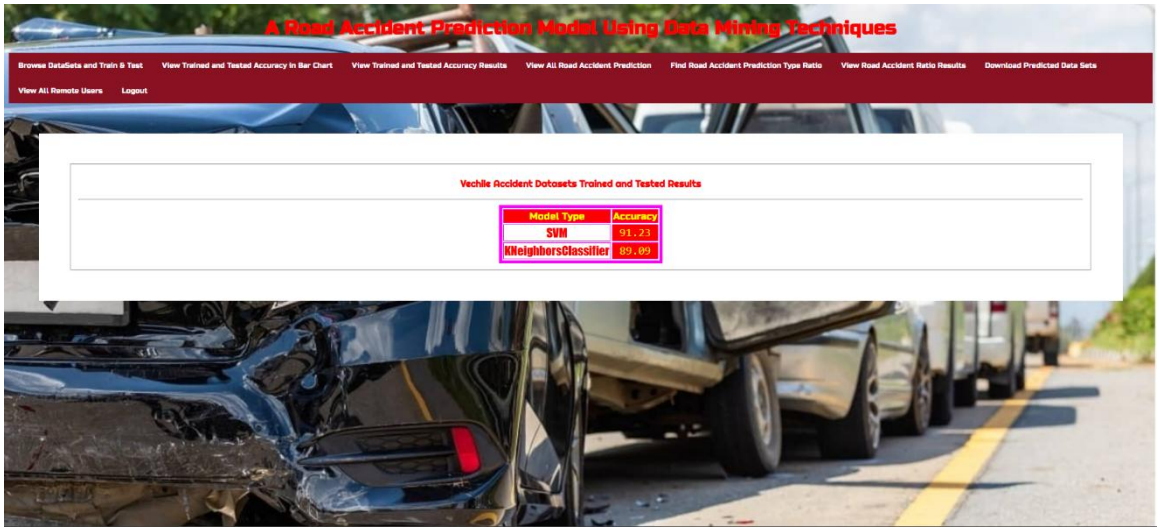


Fig. 10.7.Algorithms accuracy

Step8: Here is the accuracy bar graph.

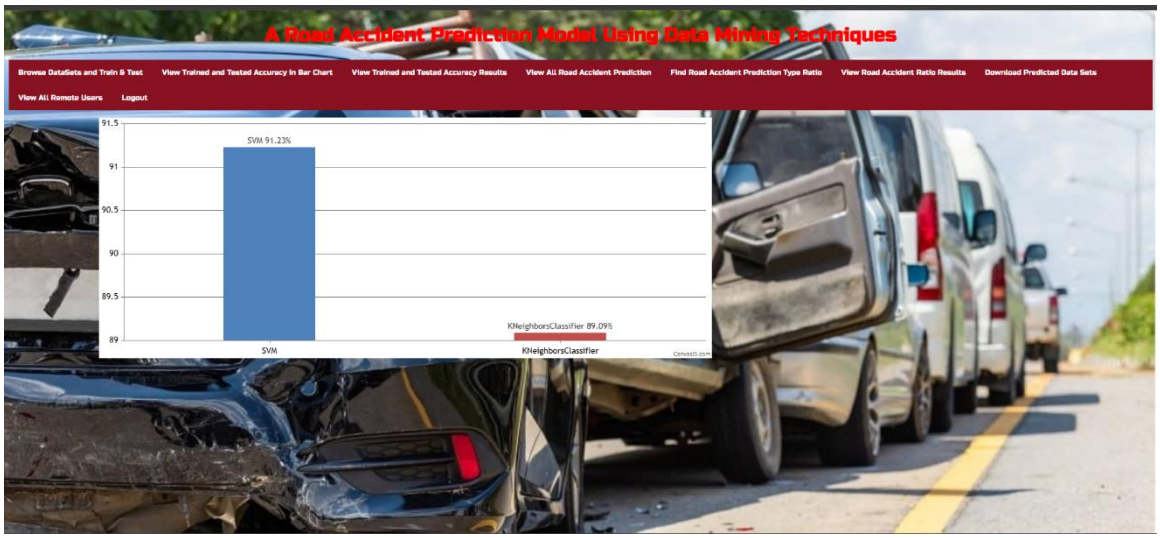


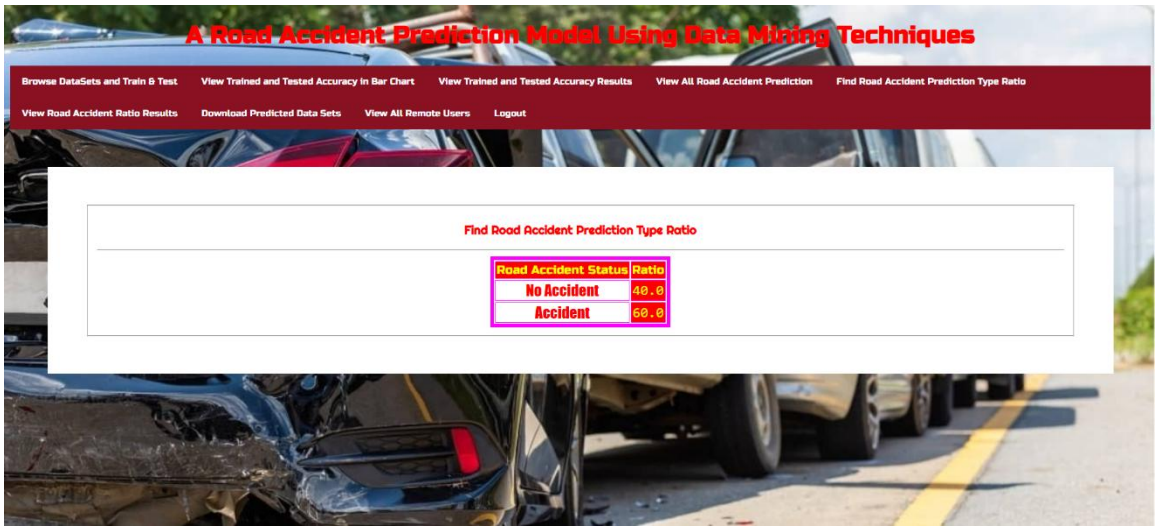
Fig. 10.8.Bar graph

Step9: Accident predict data.

Reference_Number	State	Area_Name	Traffic_Rules_Violation	Vechile_Load	Time	Road_Class	Road_Surface	Lighting_Conditions	Weather_Conditions	Person_Type	Sex	Age	Type_of_Vehicle	Prediction
110016024	Chhattisgarh	BIJAPUR	Drink and Drive	Heavy Loaded	8.30	Unclassified	Wet / Damp	Daylight: street lights present	Raining without high winds	Driver	Female	39	Car	Accident
110020394	Telangana	ADILABAD	Drink and Drive	Heavy Loaded	8.30	A	Dry	Darkness: street lighting unknown	Fine without high winds	Driver	Female	40	Car	No Accident
110030967	Andhra Pradesh	KURNOOL	Over Speeding	Heavy Loaded	7.00	Unclassified	Wet / Damp	Daylight: street lights present	Fine with high winds	Driver	Male	45	Car	Accident
110411021	Punjab	BATHINDA	Dangerous Driving	Heavy Loaded	14 40	Unclassified	Dry	Daylight: street lights present	Fine without high winds	Driver	Female	86	Car	No Accident
110030967	Andhra Pradesh	KURNOOL	Over Speeding	Heavy Loaded	7	Unclassified	Wet / Damp	Daylight: street lights present	Fine with high winds	Driver	Male	45	Car	Accident

Fig. 10.9.Accident data

Step10: Accident ratio and logout.



Road Accident Status	Ratio
No Accident	40.0
Accident	60.0

Fig. 10.10.Accident ratio

11. CONCLUSION

In conclusion, the proposed road accident prediction model utilizing data mining techniques offers a powerful and intelligent solution for enhancing road safety and reducing accident occurrences. By analysing vast and diverse datasets with advanced machine learning algorithms, the model effectively identifies potential accident risks and hotspots with higher accuracy than traditional methods. Its ability to process real-time data and adapt through continuous learning ensures that predictions remain relevant and reliable even in dynamic traffic environments. Furthermore, the system's integration with visualization tools allows for better decision-making by traffic authorities, enabling timely preventive actions. Overall, this data-driven approach not only improves traffic management and safety planning but also contributes to building smarter, safer, and more efficient transportation systems.

12. FUTURE ENHANCEMENT

In the future, the road accident prediction model can be further enhanced by incorporating real-time data from IoT-enabled traffic sensors, CCTV cameras, and vehicle telemetry systems to provide instant and dynamic risk assessments. Integration with GPS and navigation applications can offer on-the-spot accident risk alerts to drivers, helping them make safer route decisions. Additionally, implementing deep learning models such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) can improve pattern recognition in temporal and visual traffic data. The system can also be expanded to support multilingual voice alerts and mobile applications for public use. Moreover, predictive analytics can be combined with simulation tools to forecast the impact of future infrastructure changes on accident rates. Collaboration with smart city infrastructure can make the model more efficient and contribute to the development of intelligent traffic systems that respond automatically to high-risk scenarios, further minimizing accident rates and enhancing urban mobility.

REFERENCES

Beshah, T., & Hill, S. (2010). Mining road traffic accident data to improve safety: Role of road-related factors on accident severity in Ethiopia. Australasian Transport Research Forum, 1–15.

Chang, L. Y., & Chen, W. C. (2005). Data mining of tree-based models to analyze freeway accident frequency. Journal of Safety Research, 36(4), 365–375. https://doi.org/10.1016/j.jsr.2005.06.010

Sharma, A., & Mukherjee, S. (2015). Road traffic accident prediction and analysis using random forest classification. International Journal of Engineering Research and Applications, 5(10), 1–5.

Khan, A. M., & Razzak, M. I. (2019). An ensemble approach for road accident prediction using machine learning. International Journal of Advanced Computer Science and Applications, 10(5), 236–243.

Sohn, S. Y., & Lee, S. H. (2003). Data fusion, ensemble and clustering to improve the classification accuracy for the severity of road traffic accidents in Korea. Safety Science, 41(1), 1–14. https://doi.org/10.1016/S0925-7535(01)00035-7

Ramos, F. M., & Almeida, P. S. (2018). Clustering and visualization of traffic accident data: A study in Rio de Janeiro. Procedia Computer Science, 138, 355–362.

Wang, Y., &Kockelman, K. M. (2013). Forecasting the severity of crash injuries: A comparison of competing models. Accident Analysis & Prevention, 58, 469–481. https://doi.org/10.1016/j.aap.2013.06.040

Patil, B. L., & Suresh, K. (2017). Prediction and analysis of road accidents using machine learning algorithms. International Journal of Computer Applications, 169(7), 1–5.

Jureczko, M., &Madeyski, L. (2014). Using neural networks for predicting the number of road accidents. Advances in Intelligent Systems and Computing, 242, 181–190.