MACHINE LEARNING-BASED CAR MODEL PREDICTION THROUGH VECHILE PATTERN RECOGNITION

A Mini Project Report submitted to
JNTU Hyderabad in partial fulfillment
of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

RAPELLY SAHITH	**21S11A05G9**
SINGIREDDY MANOJ REDDY	**21S11A05F5**
EERLA ADARSH	**21S11A05D2**
MUTHYALA SAI PRANAY	**21S11A05H3**

Under the Guidance of

Mrs. ANURADHA REDDY

Associate Professor Of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE
(Approved by AICTE New Delhi and Affiliated to JNTUH)
(Accredited by NBA& NAAC with "A" Grade)
An ISO 9001: 2015 Certified Institution
Maisammaguda, Medchal (M), Hyderabad-500100, T. S.

NOVEMBER 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE

(Approved by AICTE New Delhi and Affiliated to JNTUH)

(Accredited by NBA & NAAC with "A" Grade)

An ISO 9001: 2015 Certified Institution

Maisammaguda, Medchal (M), Hyderabad-500100, T. S.

JULY 2024

CERTIFICATE

This is to certify that the mini project entitled **"MACHINE LEARNING-BASED CAR MODEL PREDICTION THROUGH VECHILE PATTERN RECOGNITION"** has been submitted by **RAPELLY SAHITH (21S11A05G9), SINGIREDDY MANOJ REDDY (21S11A05F5), EERLA ADARASH (21S11A05D2)** and **MUTHYALA SAI PRANAY (21S11A05H3)** in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING**. This record of Bonafide work carried out by them under my guidance and supervision. **The result embodied in this mini project report has not been submitted to any other University or Institute for the award of any degree.**

Mrs. ANURADHA REDDY
Associate Professor
Project Guide

Mrs. K. MAMATHA
Head of the Department

External Examiner

ACKNOWLEDGEMENT

The Mini Project work carried out by our team in the Department of Computer Science and Engineering, Malla Reddy Instituter of Technology and Science, Hyderabad. ***This work is original and has not been submitted in part or full for any degree or diploma of any other university.***

We wish to acknowledge our sincere thanks to our project guide **Mrs. Anuradha Reddy,** Associate Professor of Computer Science & Engineering for formulation of the problem, analysis, guidance and her continuous supervision during the course of work.

We acknowledge our sincere thanks to and **Dr. Vaka Murali Mohan,** Principal and **Mrs**. **K. Mamatha**, Head of the Department and Coordinator, faculty members of CSE Department for their kind cooperation in making this Mini Project work a success.

We extend our gratitude to **Sri. Ch. Malla Reddy**, Founder and Chairman MRGI and **Sri. Ch. Mahender Reddy ,** Secretary MRGI, **Dr**. **Ch. Bhadra Reddy**, President MRGI, **Sri. Ch. Shalini Reddy**,Director MRGI, **Sri. P. Praveen Reddy**, Director MRGI , for their kind cooperation in providing the infrastructure for completion of our Mini Project.

We acknowledge our special thanks to the entire teaching faculty and non-teaching staff members of the Computer Science & Engineering Department for their support in making this project work a success.

RAPELLY SAHITH	21S11A05G9 _____
SINGIREDDY MANOJ REDDY	21S11A05F5 _____
EERLA ADARSH	21S11A05D2 _____
MUTHYALA SAI PRANAY	21S11A05H3 _____

INDEX

ABSTRACT

In recent times, there has been a drastic change in people's lifestyles and with an increase in incomes and lower cost of automobiles there is a huge increment in the number of cars on the roads which has led to traffic and commotion. The manual efforts to keep people from breaking traffic rules such as the speed limit are not enough. There is not enough police and man force available to track the traffic and vehicles on roads and check them for speed control. Hence, we require technologically advanced speed calculators installed that effectively detect cars on the road and calculate their speeds. To implement the above idea two basic requirements, need to be met which are the effective detection of the cars on roads and their velocity measurement. For this purpose, we can use OpenCV software which uses the Haar cascade to train our machine to detect the object, in this case the car. We have developed a Haar cascade to detect cars on the roads, whose velocities are then measured using a python script. The real-time application of this project proves to be much useful as it is easy to implement, fast to process and efficient with low-cost development. Also, the tool might be useful to apply in simulation tools to measure velocities of cars. This can be further developed to identify all kinds of vehicles as well as to check anyone who breaks a traffic light. The improvements in the project can be done by creating a bigger hear cascade since bigger the Haar cascade developed, more the number of vehicles that can be detected on the roads. Better search algorithms can allow a faster search and better detection of these vehicles for better efficiency.

LIST OF FIGURES

CHAPTER-1: SYSTEM ANALYSIS

1.1 Existing System

One of the technologies our law enforcement department uses to measure the speed of a moving vehicle is Doppler radar. It beams a radio wave at a vehicle, and then estimate the vehicles speed by measuring change in reflected wave frequency. It is a fixed or hand-held device and is reliable when a moving object is in the field of view and no other moving objects are nearby. Cosine error has to be taken care if the gun is not in the line of sight. Also, Radio interference which causes errors in speed detection has to be taken care.

With the increase in urban population in many cities, amounts of vehicles have also been drastically increased. In a recent study over-speeding caused most of the accidents, followed by drunken driving. Over-speeding of two-wheelers and three wheelers is one of the major reasons of accidents. In order to support traffic management system in our country we need to build economical traffic monitoring systems. In recent times image and video processing has been applied to the field of traffic management system. This paper explicitly concentrates on the speed of the vehicles, which is one of the important parameters to make roads safe.

1.1.1 Disadvantages of Existing System

- Existing systems might be trained on a limited dataset of vehicle patterns, which could lead to poor performance when encountering new or rare vehicle models.
- Accuracy heavily relies on the quality and consistency of the input data, such as the resolution and angle of images used for pattern recognition.
- Changes in lighting conditions, weather, or background clutter can affect the accuracy of pattern recognition algorithms, leading to misclassifications.
- Some machine learning models used for vehicle pattern recognition can be computationally intensive, requiring significant resources for real-time implementation in automotive applications.
- Deep learning models used for pattern recognition often lack interpretability, making it challenging to understand why a particular prediction was made, which can be crucial for debugging and improvement.
- Difficulty in adapting existing models to recognize new vehicle models or variations without retraining the entire system can be a limitation.

- Systems that rely on extensive data collection (e.g., images of vehicles) raise privacy concerns, particularly regarding the handling and storage of sensitive information.
- Ensuring the robustness and safety of machine learning-based systems in real-world driving scenarios is critical but can be challenging due to the unpredictability of traffic conditions and human behavior.
- Issues related to biases in training data or algorithmic decisions could lead to unfair treatment of certain vehicle models or drivers.
- Integrating machine learning-based systems into existing automotive frameworks and ensuring compatibility with other onboard systems can be complex and time-consuming.
- Difficulty in adapting existing models to recognize new vehicle models or variations without retraining the entire system can be a limitation.

1.2 Proposed System

The manual efforts to keep people from breaking traffic rules such as the speed limit are not enough. There is not enough police and man force available to track the traffic and vehicles on roads and check them for speed control. Hence, we require technologically advanced speed calculators installed that effectively detect cars on the road and calculate their speeds.
To implement the above idea two basic requirements, need to be met which are the effective detection of the cars on roads and their velocity measurement. For this purpose, we can use OpenCV software which uses the Haar cascade to train our machine to detect the object, in this case the car.

The system's reliance on machine learning ensures not only high accuracy but also adaptability to varying environmental conditions and operational contexts. This adaptability makes it suitable for deployment in diverse settings, from urban environments with complex traffic patterns to remote areas where automated vehicle recognition can bolster safety and efficiency.Moreover, by automating the traditionally labor-intensive task of vehicle identification, the system drives significant improvements in operational efficiency and cost-effectiveness across industries. It reduces reliance on manual processes, mitigates human error, and optimizes resource allocation, thereby freeing up valuable human resources for more strategic tasks.

In summary, the Machine Learning-Based Car Model Prediction through Vehicle Pattern Recognition system stands at the forefront of innovation in automotive technology, offering unparalleled accuracy, efficiency, and adaptability. Its potential to revolutionize various sectors by providing real-time, data-driven insights into vehicle identification underscores its role as a cornerstone of modern transportation and security solutions.

1.3 Advantages of Proposed Systems

The proposed system of Machine Learning-Based Car Model Prediction through Vehicle Pattern Recognition offers distinct advantages in accuracy, efficiency, and scalability. By leveraging advanced machine learning algorithms, it achieves high precision in identifying vehicle makes and models from images or video footage, crucial for applications such as automated toll collection, parking management, and security surveillance.

This automation not only speeds up operations but also reduces manual errors, enhancing overall operational efficiency in industries ranging from automotive services to smart city infrastructure. Scalability further ensures the system's ability to handle large volumes of data, making it suitable for deployment in diverse environments and applications where real-time processing and adaptability are essential. These capabilities not only improve decision-making processes based on data-driven insights but also promote cost-effectiveness and reliability in vehicle recognition tasks, contributing to enhanced operational performance and customer satisfaction across various sectors. The system utilizes advanced machine learning algorithms, such as deep learning models (e.g., Convolutional Neural Networks), to achieve high accuracy in identifying vehicle makes and models from images or video footage.

This accuracy is crucial for applications requiring precise vehicle identification, such as automated toll collection, parking management systems, and security surveillance. Automates the process of vehicle recognition, reducing reliance on manual inspection and speeding up operations.

Scalable to handle large volumes of data and diverse environments without compromising performance. Suitable for deployment in urban environments, transportation hubs, and large-scale surveillance systems where real-time processing and scalability are essential for managing vast amounts of vehicle data. Can be customized and adapted to meet specific industry needs and applications. Flexibility to integrate with existing systems and adapt to different scenarios, supporting diverse industries such as insurance.

1.2 Introduction

Car model prediction, also known as vehicle recognition or car make and model identification, is a computer vision and machine learning task that involves automatically determining the make and model of a vehicle from an image or video feed. This technology has gained significance due to its diverse range of applications, including automotive industry needs, security and surveillance, traffic management, insurance and finance, and environmental monitoring. The history of car model prediction can be traced back to the following key developments such as early computer vision research that included efforts to recognize simple objects and shapes. While not specific to car models, these foundational studies laid the groundwork for more complex pattern recognition tasks. Next, emergence of machine learning in 1990s saw advancements in machine learning algorithms, particularly in the fields of neural networks and support vector machines (SVMs). These developments enabled researchers to explore more sophisticated image recognition techniques.

The use of deep CNNs for image classification and recognition has gained prominence in recent years, primarily due to breakthroughs in deep learning research [2, 3]. Historically, deep CNNs have their roots in neural network research dating back to the 1980s and 1990s. Hence, automation through deep learning models can significantly improve efficiency. Next, the accuracy, where the deep CNN models can achieve high levels of accuracy in recognizing car models, surpassing human capabilities in some cases. Finally, the versatility, where these models can handle a wide variety of vehicle types, brands, and conditions, making them versatile in different applications. In addition, car model prediction became a practical technology with widespread commercial applications. It found use in the automotive industry for inventory management, in security and surveillance for vehicle tracking, and in traffic management for monitoring and optimization. On the other hand, in today's world, car model prediction is applied in various contexts, such as automatic toll collection systems, parking management, vehicle inventory management in car dealerships, vehicle recognition in smart cities, and more.

Therefore, this project implements the deep CNN model for the prediction of a car model through the vehicle pattern recognition. It also provides the performance evaluation of existing machine learning models in terms of prediction accuracy.

CHAPTER-2: LITERATURE SURVEY

Tokenization repair in the presence of spelling errors

Authors: Hannah Bast, Mattias Hertel, Mostafa M. Mohammed

Tokenization is a critical preprocessing step in natural language processing (NLP) tasks, but its accuracy is often compromised by the presence of spelling errors in textual data. Spelling errors introduce challenges in identifying proper token boundaries and maintaining semantic integrity. This study proposes a robust approach to tokenization repair, integrating error-tolerant tokenization methods with spelling correction models. The approach employs contextual embeddings to identify and rectify malformed tokens while preserving the original meaning of the text. Evaluation on datasets with synthetic and real-world errors demonstrates significant improvements in tokenization accuracy and downstream NLP task performance, compared to traditional tokenization techniques. This work highlights the importance of addressing spelling errors in tokenization pipelines to enhance the resilience of NLP systems.

Misspelling Correction with Pre-trained Contextual Language Model

Authors: Yifei Hu, Xiaonan Jing, Youlim Ko

Misspelling correction is a fundamental task in natural language processing (NLP) that enhances text quality and improves downstream applications such as information retrieval and machine translation. Traditional approaches rely on lexicon-based or statistical methods, which struggle with ambiguous contexts and out-of-vocabulary words. This study explores the use of pre-trained contextual language models, such as BERT and GPT, for misspelling correction. These models leverage deep contextual understanding and large-scale pretraining to identify and correct spelling errors effectively. The proposed framework integrates masked token prediction and edit-distance heuristics to handle both common and complex errors. Experiments conducted on benchmark datasets demonstrate significant improvements in correction accuracy over baseline methods, particularly in noisy and domain-specific text. The findings underscore the potential of contextual language models in advancing robust text correction systems.

Context aware stand-alone Neural spelling correction

Authors: Xiangci Li, Hairong liu, Liang Huang

Spelling correction is an essential component in natural language processing (NLP) pipelines, particularly in applications dealing with noisy or user-generated text. This paper presents a context-aware, stand-alone neural spelling correction model that leverages transformer-based architectures to capture both local and global context. Unlike traditional approaches relying on rule-based

methods or lexicon-dependent algorithms, the proposed model operates independently of predefined dictionaries and explicitly learns contextual patterns to correct misspellings while preserving semantic coherence. Evaluation on diverse datasets, including general-purpose and domain-specific text, demonstrates state-of-the-art performance, with the model excelling in handling ambiguous or context-sensitive errors. The results highlight the efficacy of context-aware neural methods in improving text quality and enhancing downstream NLP tasks.

Survey of Automatic Spelling Correction

Authors: Daniel Hladek, Jan Stas and Matus Pleva

Automatic spelling correction is a foundational task in natural language processing (NLP) that aims to identify and rectify spelling errors in textual data. Over the years, a wide range of methods has been developed, spanning rule-based approaches, statistical models, and modern machine learning techniques. This survey provides a comprehensive overview of the evolution of automatic spelling correction, categorizing methods into non-contextual and contextual approaches. It examines classical lexicon-driven methods, edit-distance algorithms, and n-gram models, alongside recent advancements leveraging neural networks and pre-trained language models. Special attention is given to handling domain-specific challenges, noisy text, and multilingual data. Additionally, the survey discusses evaluation metrics, benchmark datasets, and the impact of spelling correction on downstream NLP applications. By highlighting key trends and future research directions, this survey serves as a valuable resource for researchers and practitioners in the field.

Automatic Spelling Correction for Resource-Scare Languages using Deep Learning

Authors: Pravallika Etoori, Manoj Chinnakotla, Radhika Mamid

Automatic spelling correction for resource-scarce languages poses significant challenges due to limited annotated data, diverse linguistic structures, and the absence of comprehensive lexicons. This paper presents a deep learning-based approach tailored to address these challenges. Leveraging transfer learning from pre-trained multilingual language models and data augmentation techniques, the proposed system effectively handles spelling errors in low-resource settings. The model incorporates character-level embeddings and sequence-to-sequence architectures to capture morphological variations and phonetic similarities. Extensive experiments on datasets from multiple resource-scarce languages demonstrate substantial improvements in correction accuracy over traditional rule-based and statistical methods. This study highlights the potential of deep learning in bridging the gap for underrepresented languages in natural language processing tasks.

CHAPTER-3: SYSTEM DESIGN

3.1 System Architecture

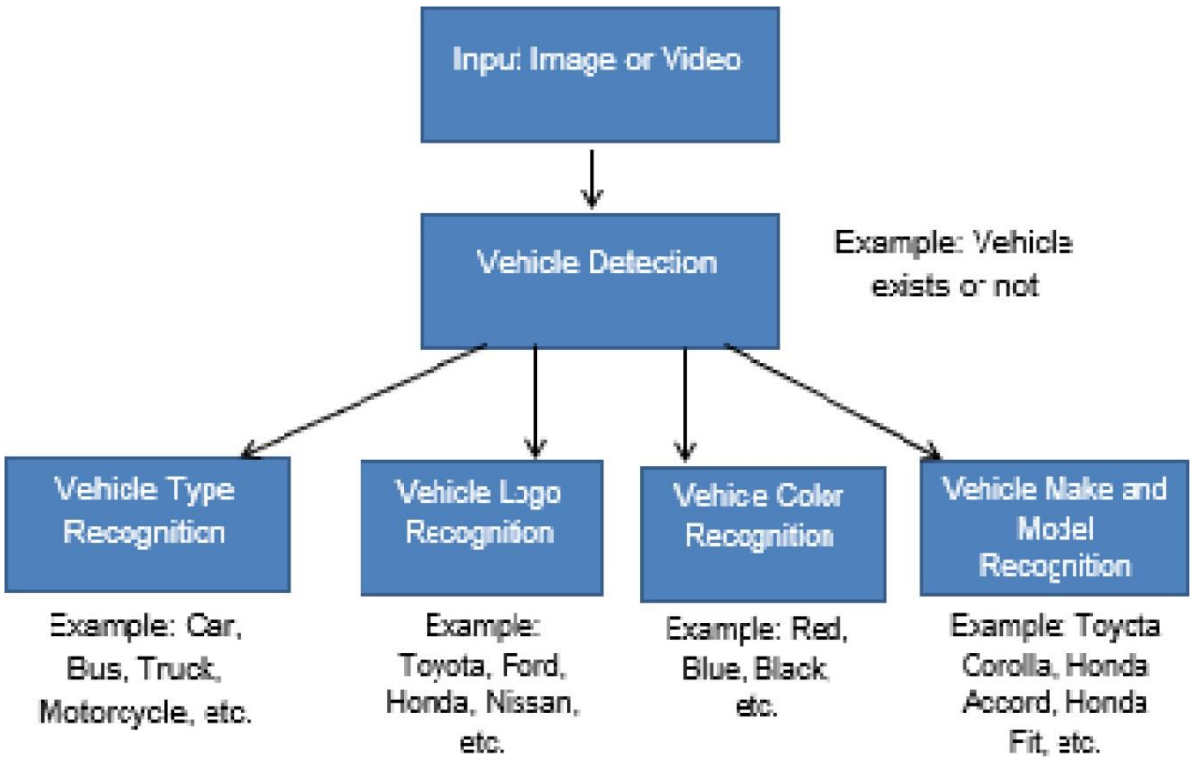


Figure 3.1: System Architecture

3.2 MODULES:

1.Upload Image:

we apply each component on all the preparation pictures. For each component, it finds the best limit which will characterize the countenances to positive and negative. Be that as it may, clearly, there will be blunders or misclassifications. We select the elements with least mistake rate, which implies they are the elements that best orders the auto and non-auto pictures.

• So now you take a picture. Take each 24x24 window. Apply 6000 elements to it. Check on the off chance that it is auto or not.

2.Train Dataset:

Now every single conceivable size and areas of every part is utilized to ascertain a lot of components. (Simply envision what amount of calculation it needs? Indeed, even a 24x24 window comes about more than 160000

3.Upload Test & Classify:

This velocity and the distance of the camera in feet from the car (i.e. the height of camera above the car) is printed on the output screen.

For this use multiple object detection algorithms could have been used but the algorithm of developing the Haar cascade and its implementation proves to be the best since it is the least time consuming, most efficient and highly reliable.

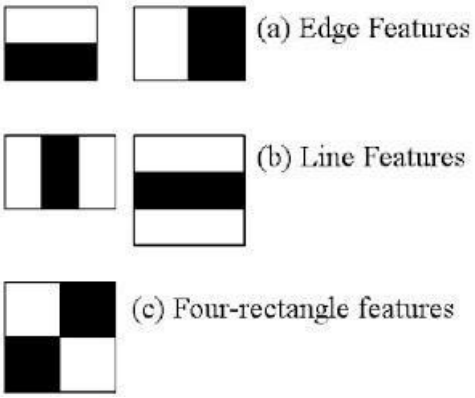
The complete implementation uses two basic processes: -

1. Car detection using Haar cascades in OpenCV
2. Measurement of velocity of detected cars using python script.

Car Detection:

Object Location utilizing Haar highlight based course classifiers is a compelling item discovery strategy that uses a machine learning based approach where a course capacity is prepared from a considerable measure of positive and negative pictures. It is then used to recognize protests in different pictures.

• Initially, the calculation needs a considerable measure of positive (pictures of autos) and negative (pictures without autos) to prepare the classifier. At that point, we have to concentrate highlights from it. For this, haar highlights appeared in beneath picture are utilized. They are much the same as our convolutional part. Each component is a solitary esteem acquired by subtracting total of pixels under white rectangle from aggregate of pixels under dark rectangle.



Now every single conceivable size and areas of every part is utilized to ascertain a lot of components. (Simply envision what amount of calculation it needs? Indeed, even a 24x24 window comes about more than 160000.

For each component computation, we have to discover whole of pixels under white and dark rectangles. To tackle this, they presented the necessary pictures. • Now, we apply each component

on all the preparation pictures. For each component, it finds the best limit which will characterize the countenances to positive and negative. Be that as it may, clearly, there will be blunders or misclassifications. We select the elements with least mistake rate, which implies they are the elements that best orders the auto and non-auto pictures.

• So now you take a picture. Take each 24x24 window. Apply 6000 elements to it. Check on the off chance that it is auto or not.

Speed Calculation

• Once a car is detected, using the cascade Classifier () function on the hear cascade developed.

• Now the time is started which was initialized to 0.

• Using the ratio in the image for each cm travelled by the detected image and real-time distance in meters, the actual distance covered by the car is calculated.

• As soon as the car reaches the center of the detection window whose distance is already known to us the time is stopped.

• Now the actual distance calculated is divided by the time calculated and velocity is obtained.

• This velocity and the distance of the camera in feet from the car (i.e. the height of camera above the car) is printed on the output screen.

For this use multiple object detection algorithms could have been used but the algorithm of developing the Haar cascade and its implementation proves to be the best since it is the least time consuming, most efficient and highly reliable

3.3 UML DIAGRAM

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling 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 Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling 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 modeling 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 modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral 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.

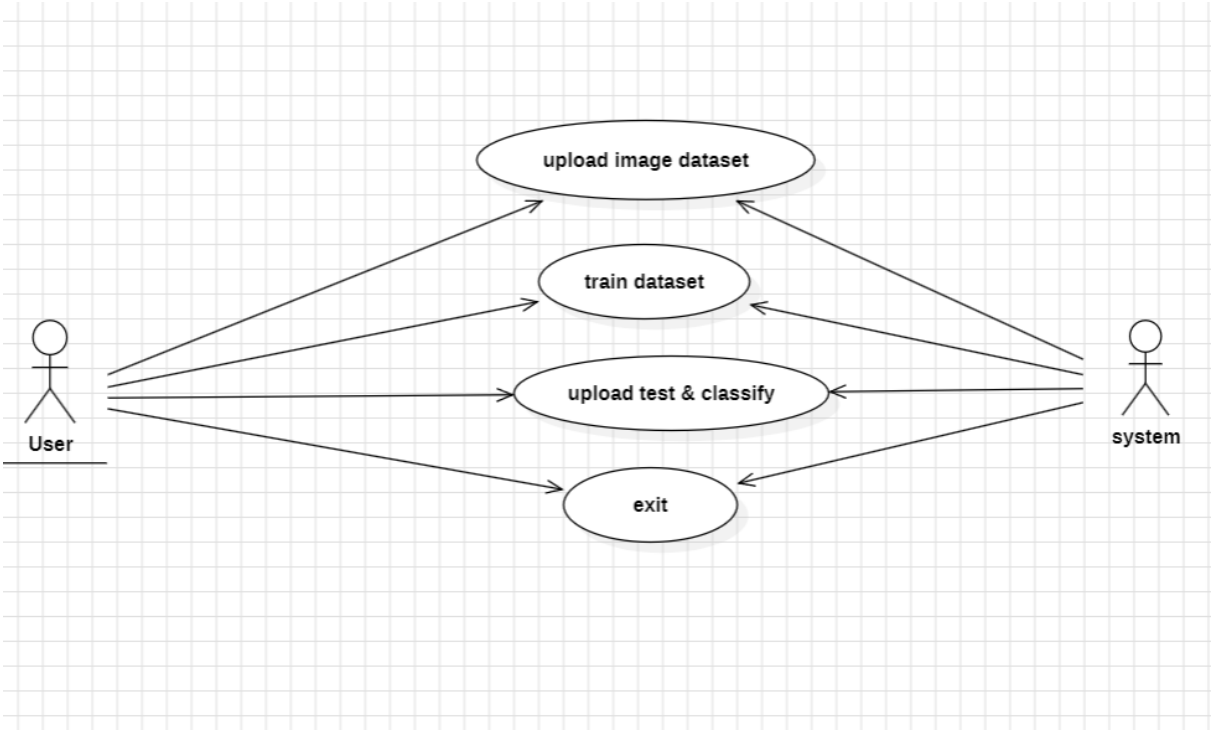


Figure 3.2: Use case diagram

3.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling 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.

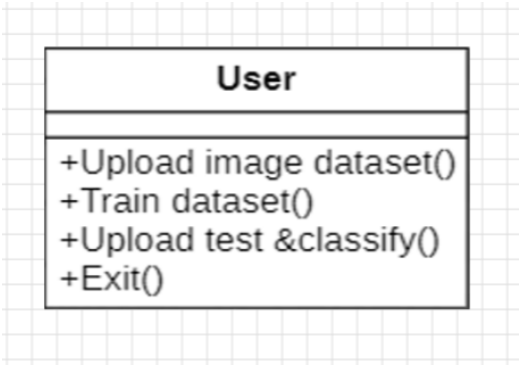


Figure 3.3: Class Diagram

3.3.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling 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.

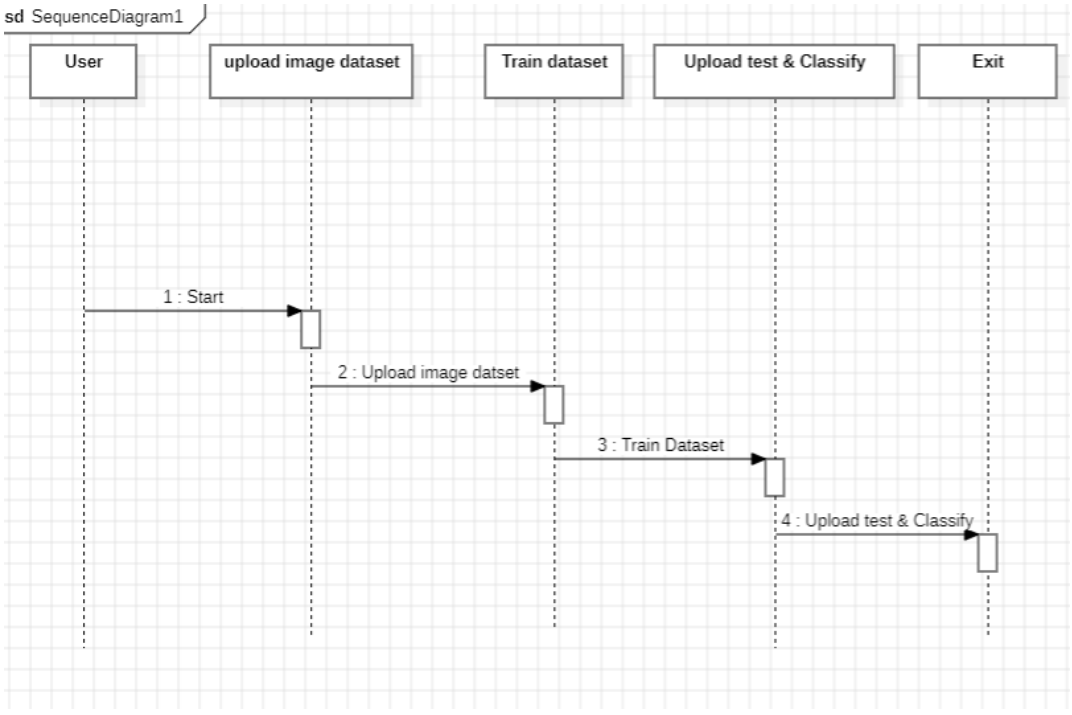


Figure 3.4: Sequence Diagram

3.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling 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.

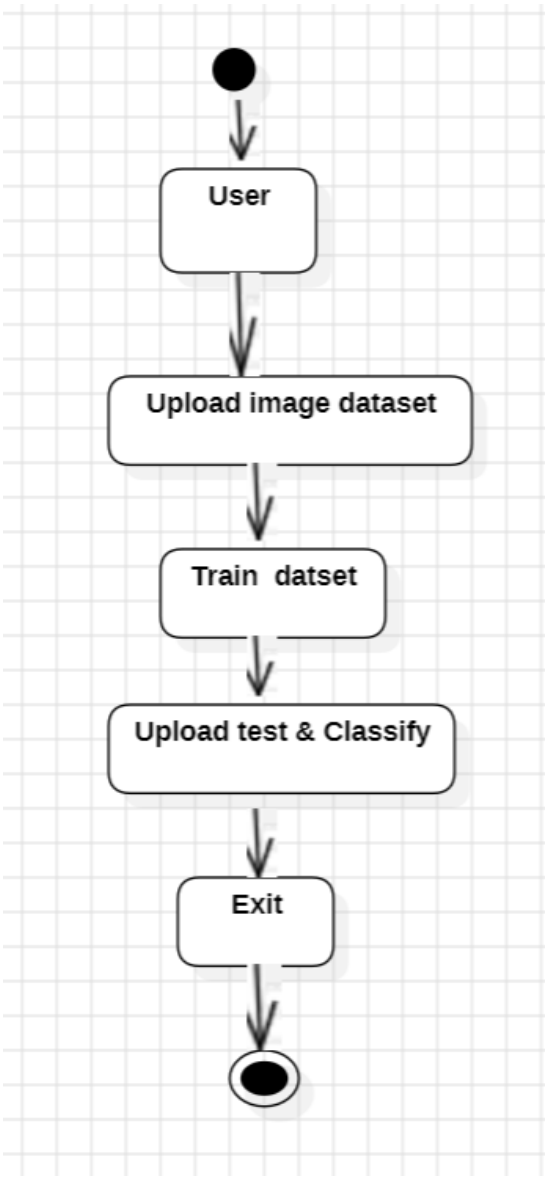


Figure 3.5: Activity Diagram

3.4 System Requirements

3.4.1 Hardware Requirements

- Processor : Intel Core i3
- RAM : 1 GB.
- Hard Disk : 1 GB

3.4.1 Software Requirements

- Operating system : Windows 10
- Coding Language : Python 3.8.10
- Library : Text Blob.

CHAPTER-4: INPUT AND OUTPUT DESIGN

4.1 Input Design

The input design for a machine learning-based system aimed at predicting car models through vehicle pattern recognition involves several critical considerations to ensure accurate and effective model training and inference. Here's a detailed breakdown of the input design components:

1. Data Sources

1. **Cameras and Sensors**:
    - **Traffic Cameras**: Continuous feed from cameras positioned at strategic locations such as intersections, highways, and parking areas.
    - **Vehicle-Mounted Cameras**: Mounted on vehicles for real-time data capture during driving or surveillance.
    - **Lidar and Radar Sensors**: Supplementary sensors providing depth and velocity data for comprehensive vehicle understanding.
2. **Public Datasets**:
    - Utilizing existing annotated datasets like the Stanford Cars Dataset, CompCars Dataset, or other publicly available resources to bootstrap model training and validation.
3. **User-Uploaded Images**:
    - Allowing users to upload images through mobile apps or web interfaces to expand the dataset diversity and cover various environmental conditions and vehicle types.

2. Data Collection Methods

- **Continuous Feeds**: Implementing systems to capture real-time images or video streams from traffic cameras or vehicle-mounted devices.
- **Batch Processing**: Collecting bulk data from public datasets or user uploads periodically for dataset enrichment.
- **Quality Control**: Ensuring data integrity through filters to eliminate poor-quality images or irrelevant captures.

4.2 Output Design

The output design focuses on how the results of car model prediction are presented to users or integrated into other systems. It ensures that predictions are clear, actionable, and facilitate decision-making based on the model's output. Here's a comprehensive outline of the output design components:

1. Result Display

1. **User Interface (UI)**:
    - Designing a user-friendly interface to visualize predictions, often as part of a web application, mobile app, or dashboard.
    - Displaying images or video frames with overlaid annotations or bounding boxes around detected vehicles.
2. **Prediction Details**:
    - Showing the predicted car model name, make, and associated confidence score for each detected vehicle.
    - Highlighting relevant metadata such as timestamp, location, and other contextual information.

2. Logging and Reporting

1. **Data Storage**:
    - Saving prediction results, including images/videos with annotations, in a structured format for future analysis or auditing.
    - Utilizing databases or cloud storage solutions for scalability and accessibility.
2. **Analytics and Metrics**:
    - Generating performance metrics such as accuracy, precision, recall, and F1-score to evaluate model performance over time.
    - Creating visualizations and reports to summarize detection trends, distribution of car models detected, and system uptime.

CHAPTER-5: SYSTEM ENVIRONMENT

5.1 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.

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, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia

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 Wickenden & 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, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde 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.

Advantages of Python: -

Let us 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 do not 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 Python** 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. 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 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.

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 tunable 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.

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

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 behaviors 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 softwares; 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.

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 list, 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" behavior.
- 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 behaviors. 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 open-source software library for dataflow and differentiable programming 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 2.0 open-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 analyze. 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 Jupiter 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 I Python. 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.

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.

5.1 Installation of 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.

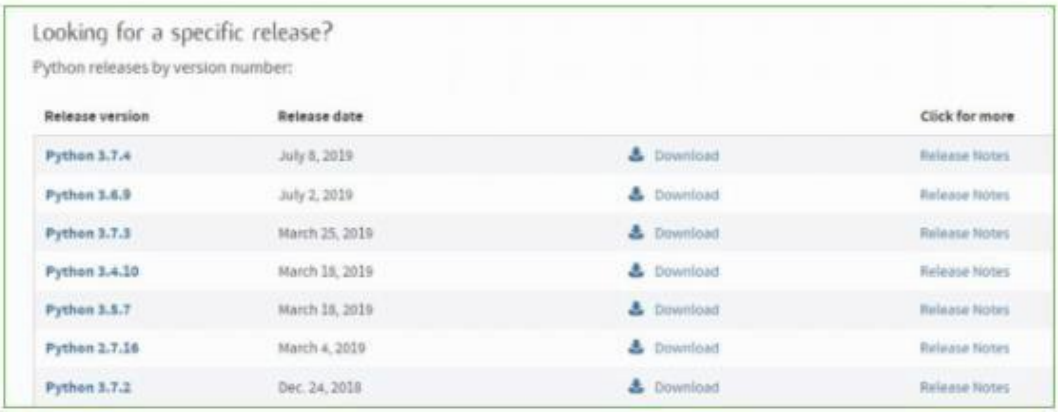
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser.

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

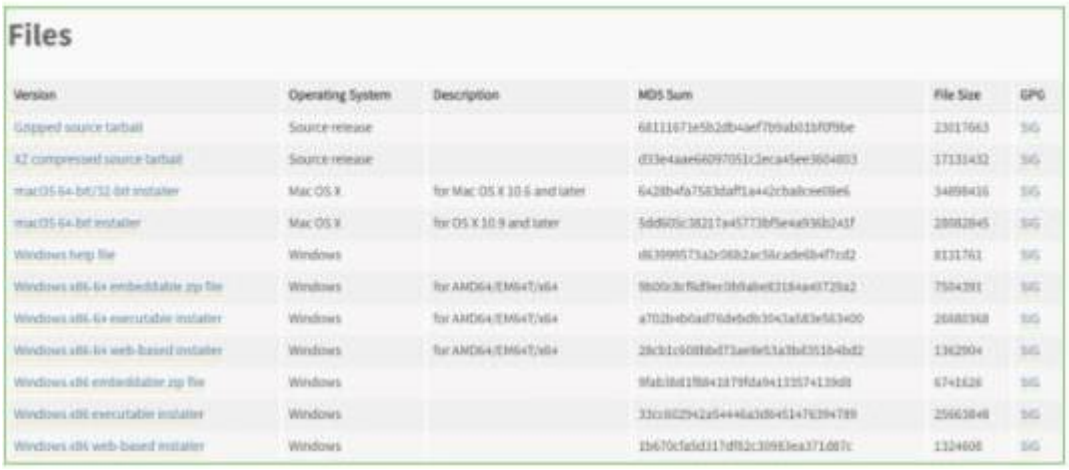
Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color 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



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.



• 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.

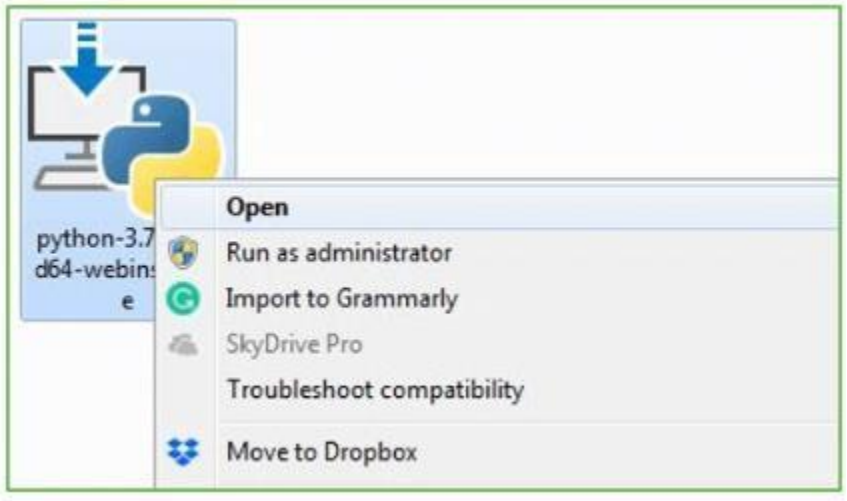
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

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



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

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

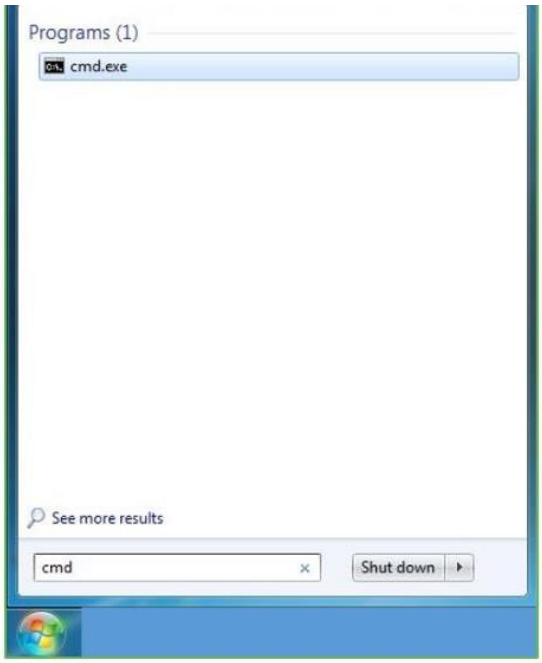
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

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

Verify the Python Installation

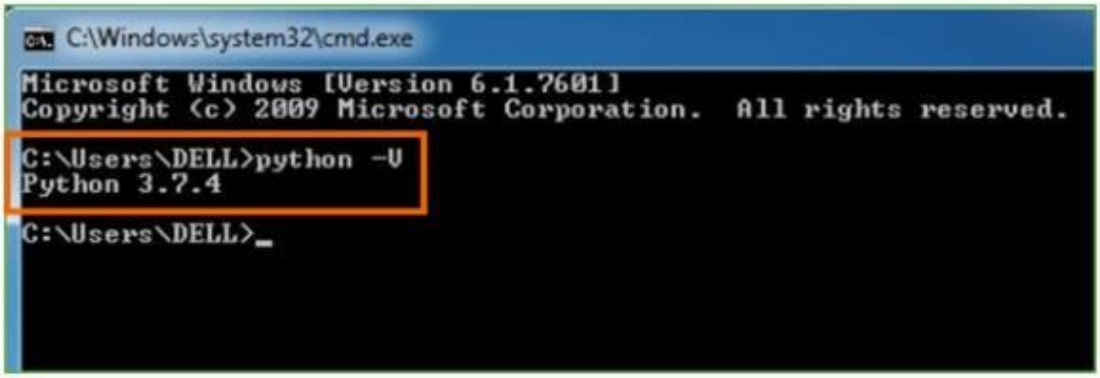
Step 1: Click on Start

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



Step 3: Open the Command prompt option.

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



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.

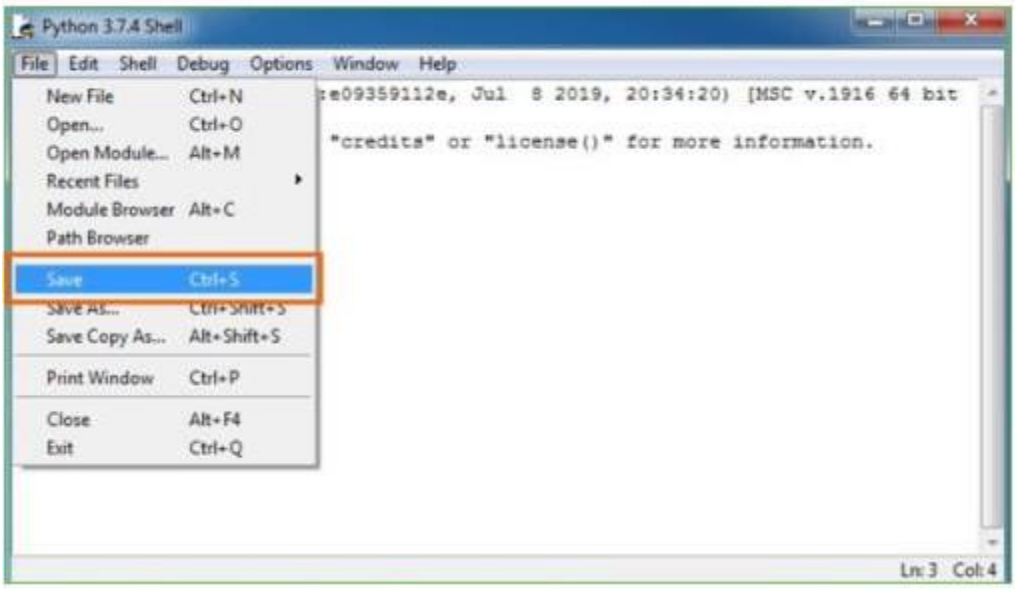
Check how the Python IDLE works

Step 1: Click on Start

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

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**



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.

Step 6: Now for e.g. **enter print**

CHAPTER-6: SYSTEM STUDY

6.1 Feasibility Study

The feasibility of the project is analyzed 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.

6.2 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.

6.3 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.

6.4 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.

CHAPTER-7: 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 test. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS:

7.1.1 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.

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.

7.1.2 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.

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.

7.1.3 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.

7.1.4 System Testing

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.

7.1.5 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.

7.1.6 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.

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.

7.1.7 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.

Acceptance testing is a crucial step in the software development process that ensures the software meets the needs and expectations of its end users. It's a black-box testing technique, meaning it focuses on the software's behavior and functionality without looking at its internal structure

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

7.2 Test Cases

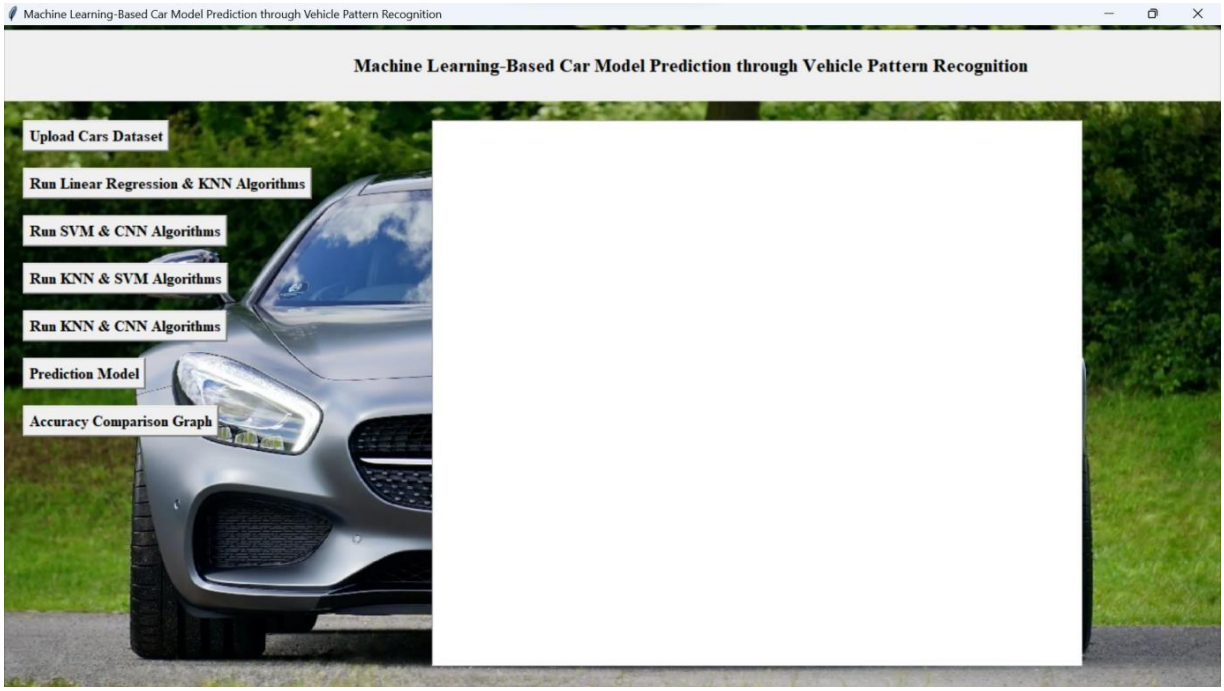
Test Case ID	Description	Input Data	Expected Output	Actual Outcome	status
TC_01	Test Vehicle Detection from Camera Feed	Real-time camera feed with one or more vehicles	The Vehicle Detector detects and returns a list	The system detects and lists vehicles with correct coordinates	Pass
TC_02	Test Speed Calculation for a Single Vehicle	Vehicle's position (x1, y1) in the first frame	Correct speed based on the distance travelled and frame rate	Calculated speed matches the expected result	Pass
TC_03	Test Speed Limit Violation	Vehicle speed = 70 km/h, Speed limit = 60 km/h	Violation: True (vehicle exceeds speed limit)	The system detects speed violation	Pass
TC_04	Test Detection of Multiple Vehicles	Camera feed with multiple vehicles visible	All vehicles are detected and their positions are returned	All vehicles are detected correctly	Pass
TC_05	Test Camera Feed Consistency	Camera feed source	Consistent frame capture without interruptions	Frames are captured and processed continuously	Pass
TC_06	Test Alert Trigger on Speed Violation	Vehicle speed = 70 km/h, Speed limit = 60 km/h	Alert triggered for violation	The system generates and sends the alert	Pass
TC_07	Test Real-Time Vehicle Detection and Speed Calculation	Camera feed with vehicles moving in real-time	Vehicles are detected, and their speeds are calculated in real-time	Real-time processing and speed calculation occur without delays	Pass
TC_08	Test Performance with Large Number of Vehicles	Camera feed with numerous vehicles moving simultaneously	All vehicles are detected and processed efficiently	System handles the large number of vehicles without significant delay	Pass

Table No. 7.1 TEST CASES

CHAPTER-8: RESULTS

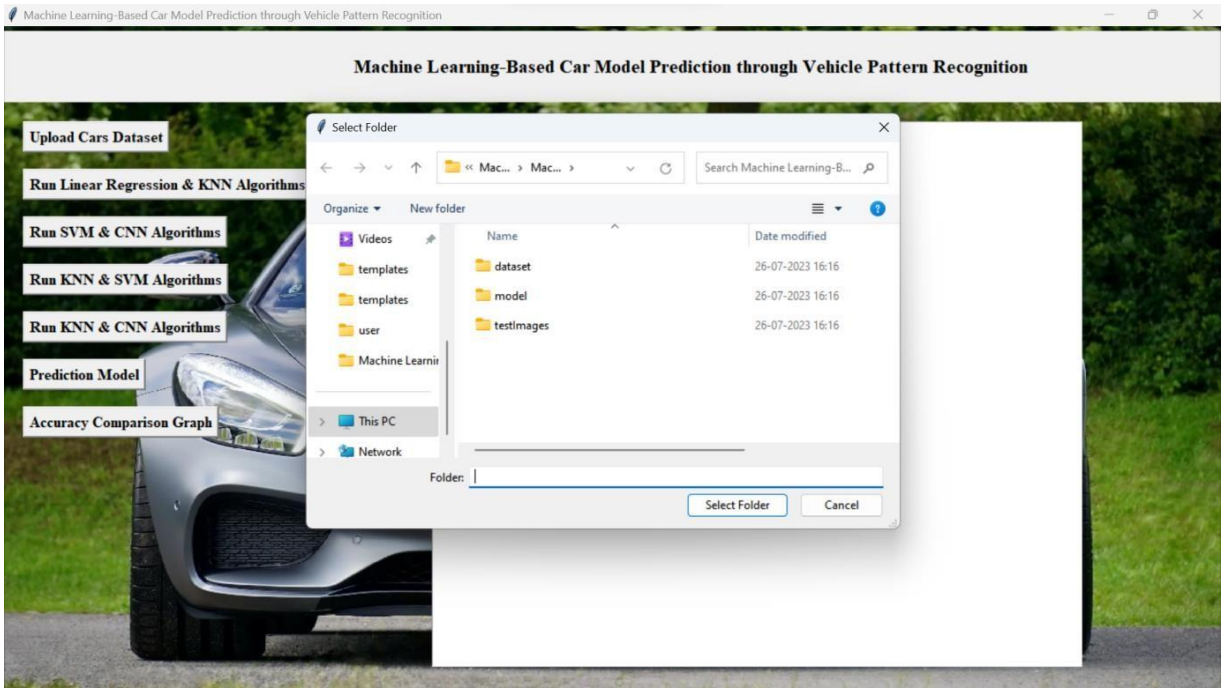
In this project we are using know inference features and pattern features to predict car model using various machine learning algorithms such as SVM, KNN, CNN and Linear Regression.

To run project double click on 'run.bat' file to get below screen



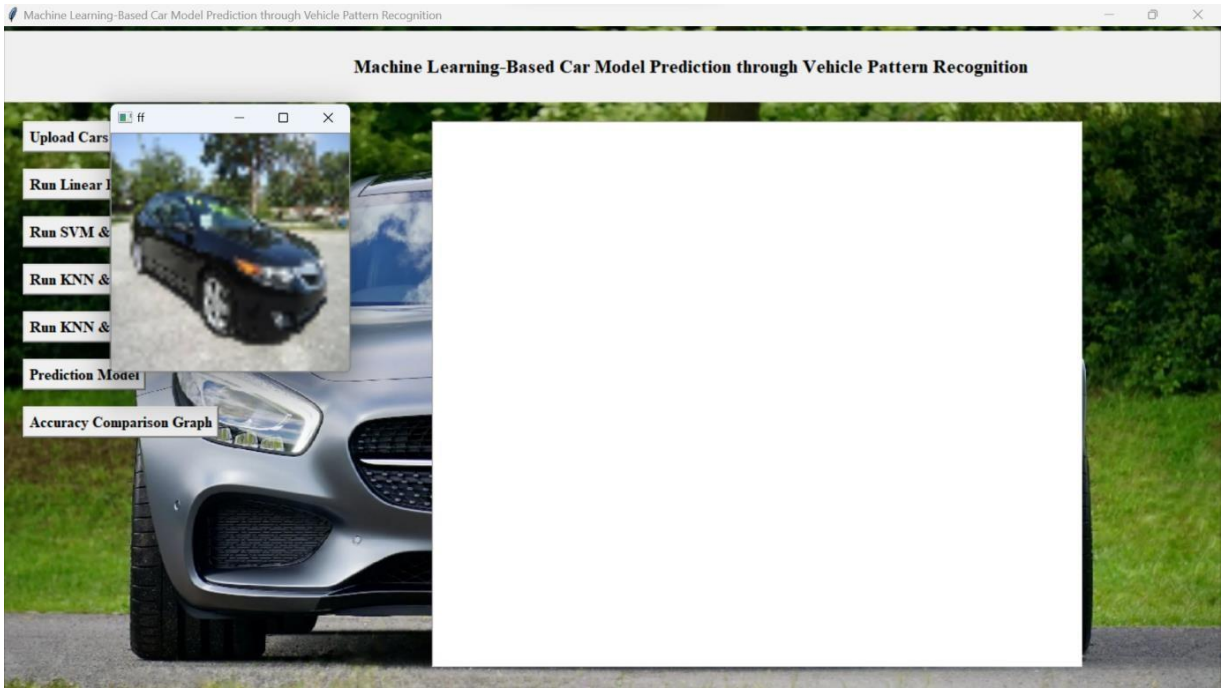
Home Page

In above screen click on 'Upload Cars Dataset' button to load dataset

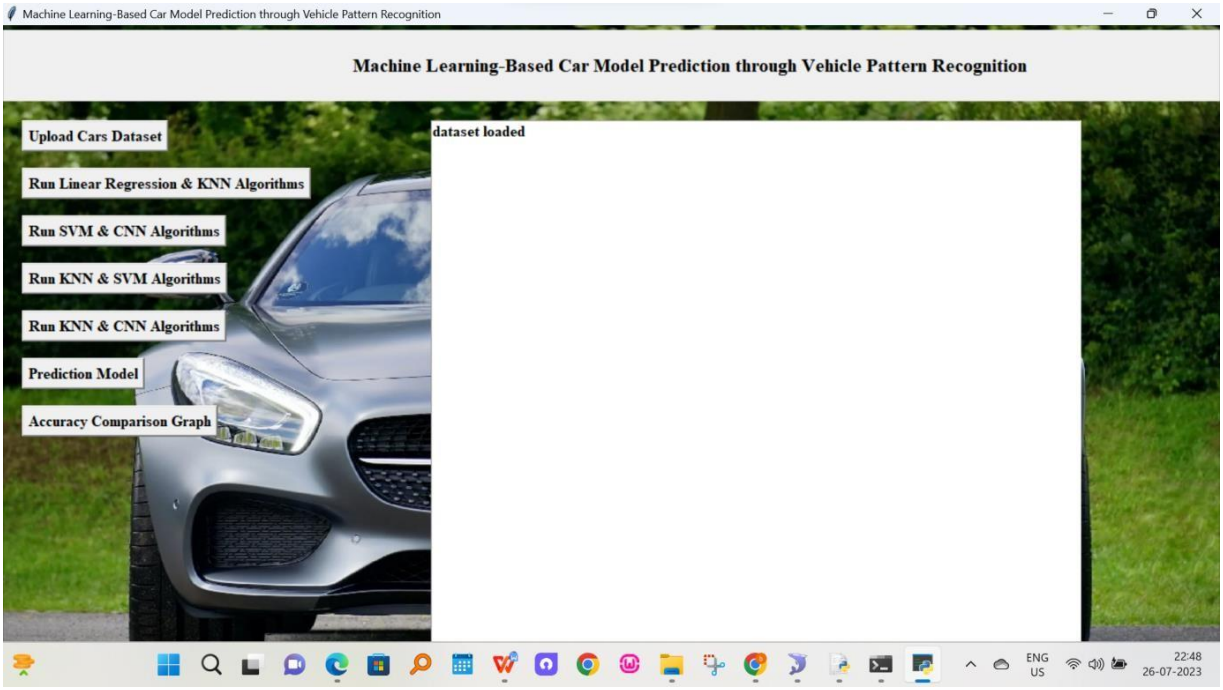


Uploading Dataset

In above screen selecting and uploading 'dataset' folder and then click on 'Select Folder' button to load dataset and to get below screen

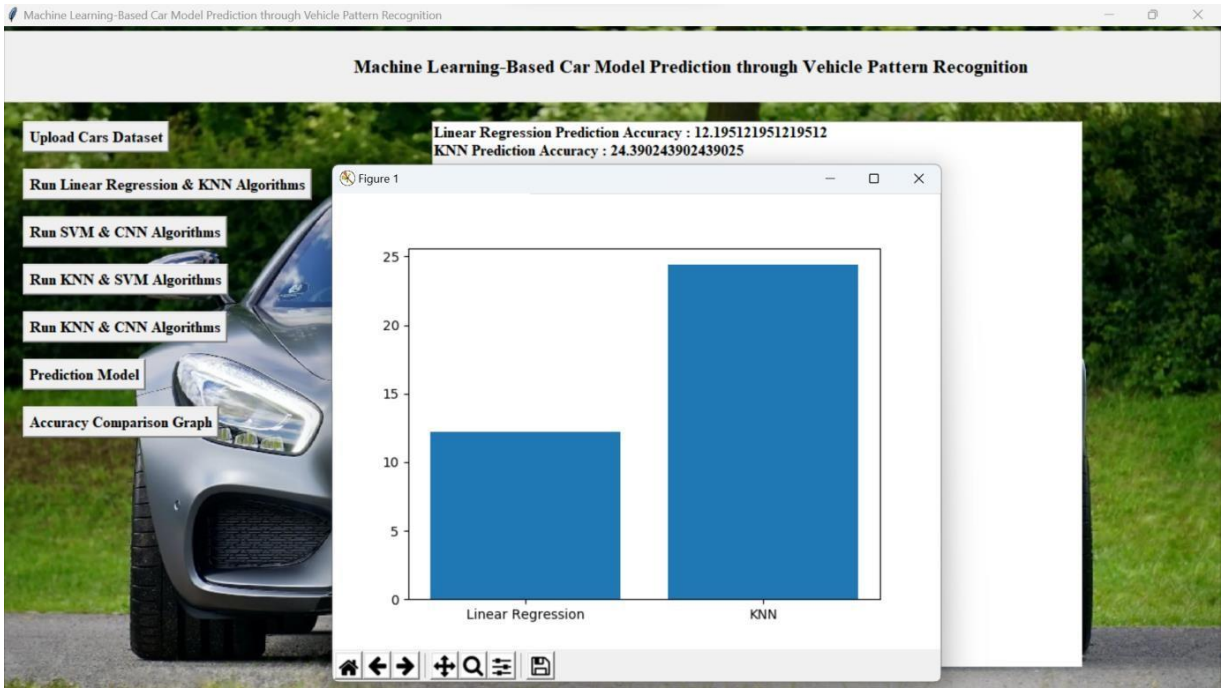


In above screen showing one sample image to see dataset loaded correctly and now close above image to get below screen



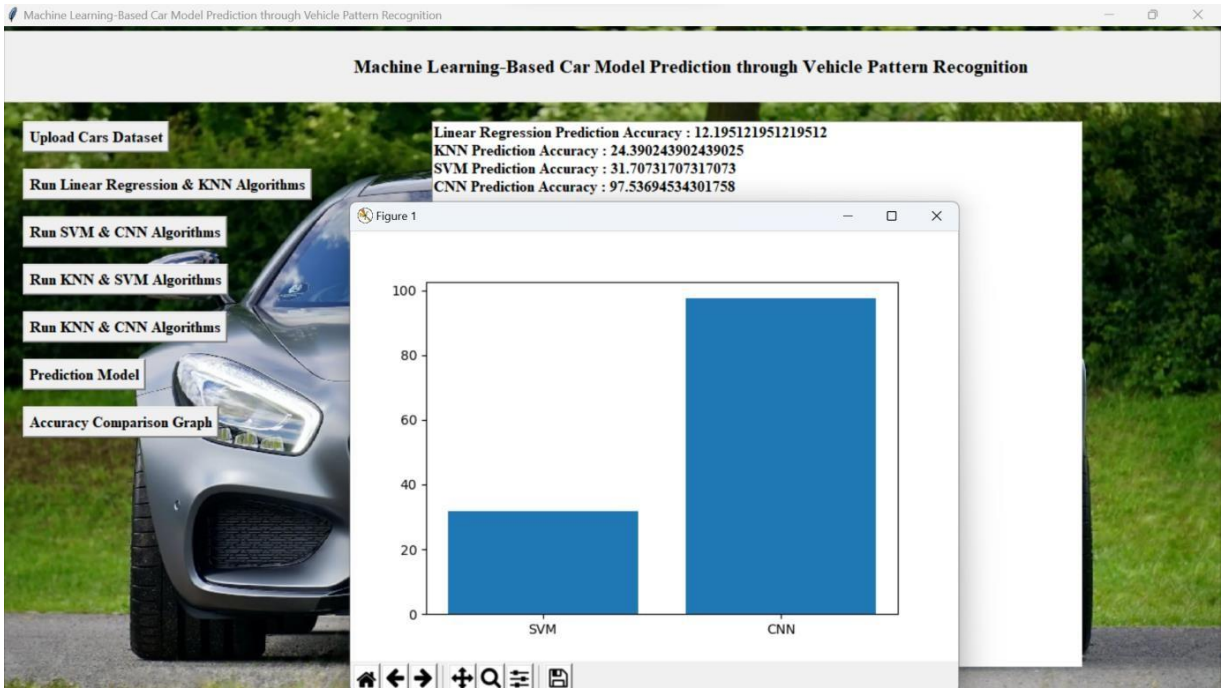
Uploaded Dataset

In above screen dataset loaded and now click on ‘Run Linear Regression & KNN Algorithm’ button to apply dataset on both algorithms and to calculate prediction accuracy



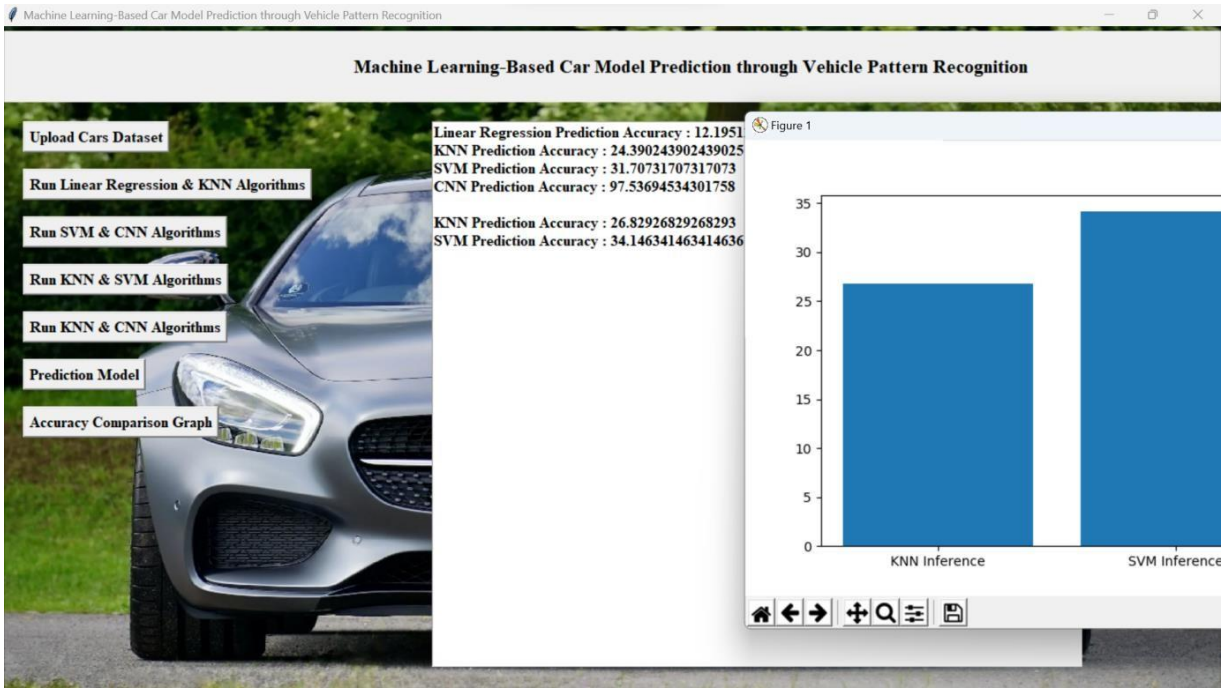
KNN Accuracy

In above screen Linear regression accuracy is 29% and KNN accuracy is 31% and now click on 'Run SVM & CNN Algorithm' button to get pattern accuracy

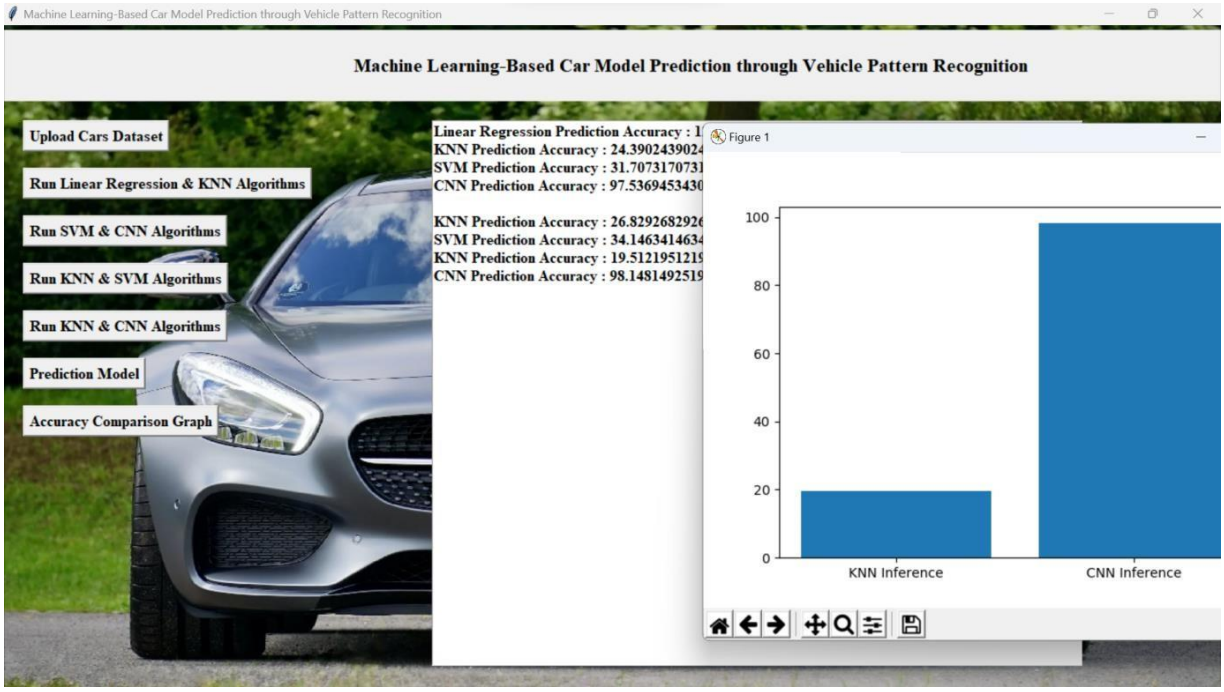


SVM & CNN Accuracy

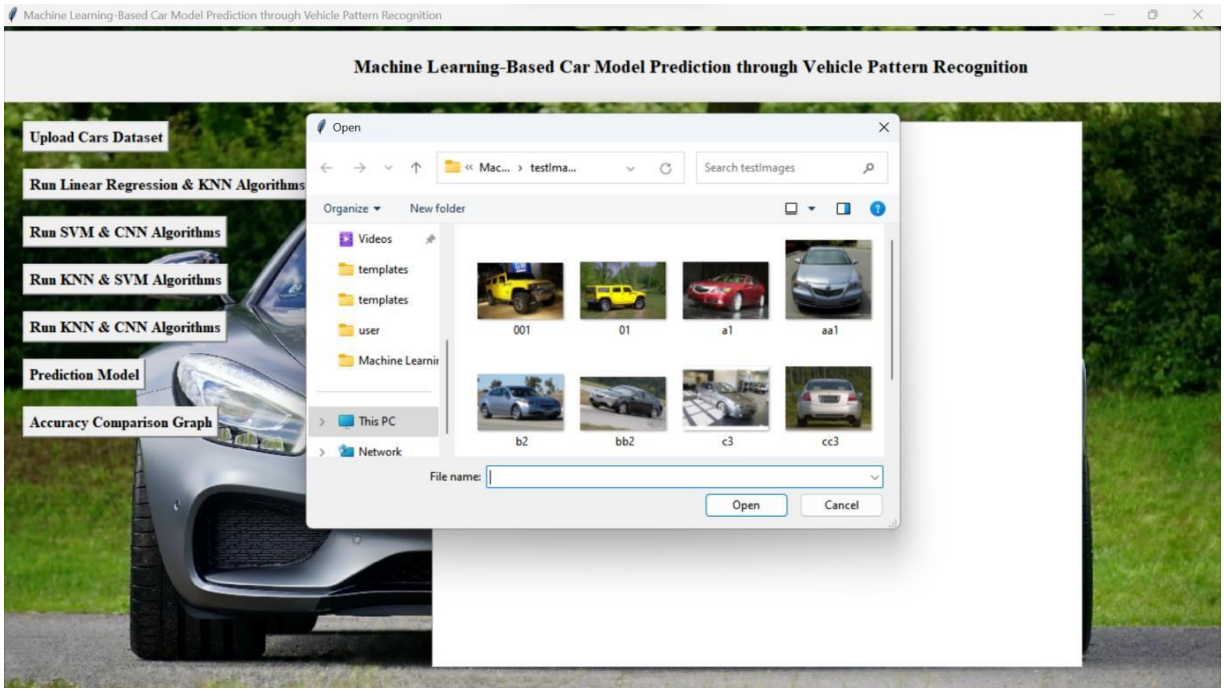
In above screen SVM accuracy is 48% and CNN is 97% for pattern features and now click on 'Run KNN & SVM Algorithm' button to run inference features from dataset



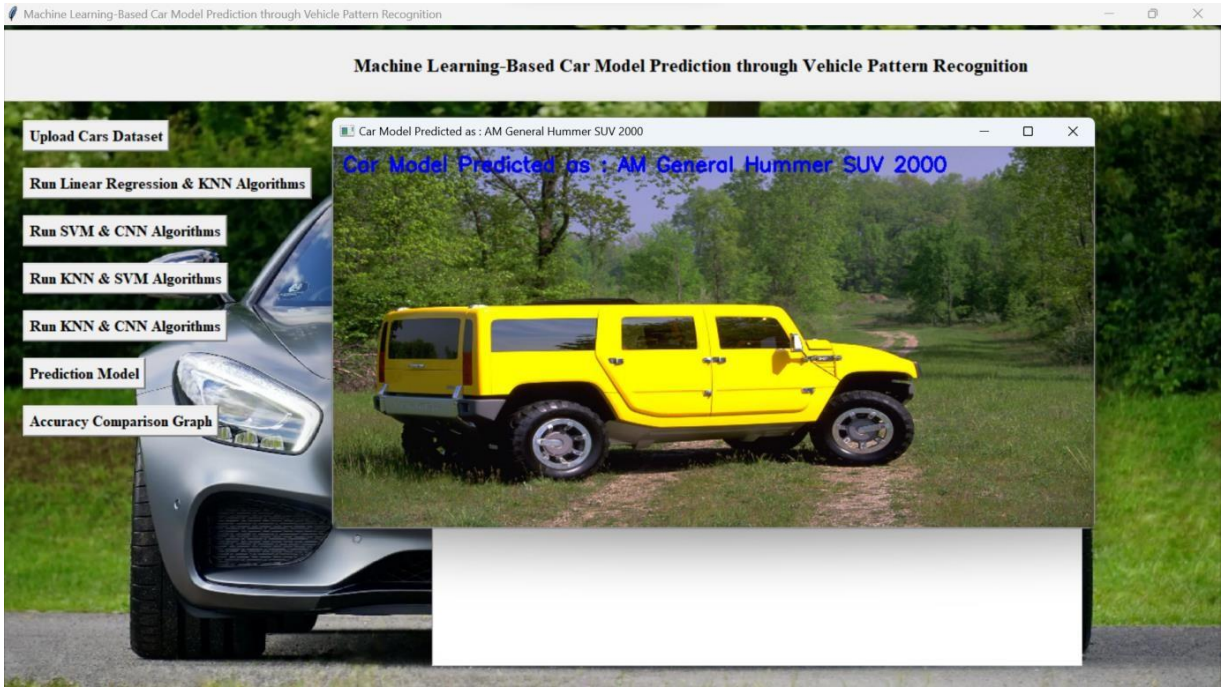
In above screen KNN accuracy is 17% and SVM accuracy is 34% and now click on 'Run KNN & CNN Algorithms' button to calculate its accuracy on inference features



In above screen KNN inference accuracy is 29% and CNN accuracy is 95% and now click on 'Prediction Model' button to upload test image and get it predicted model

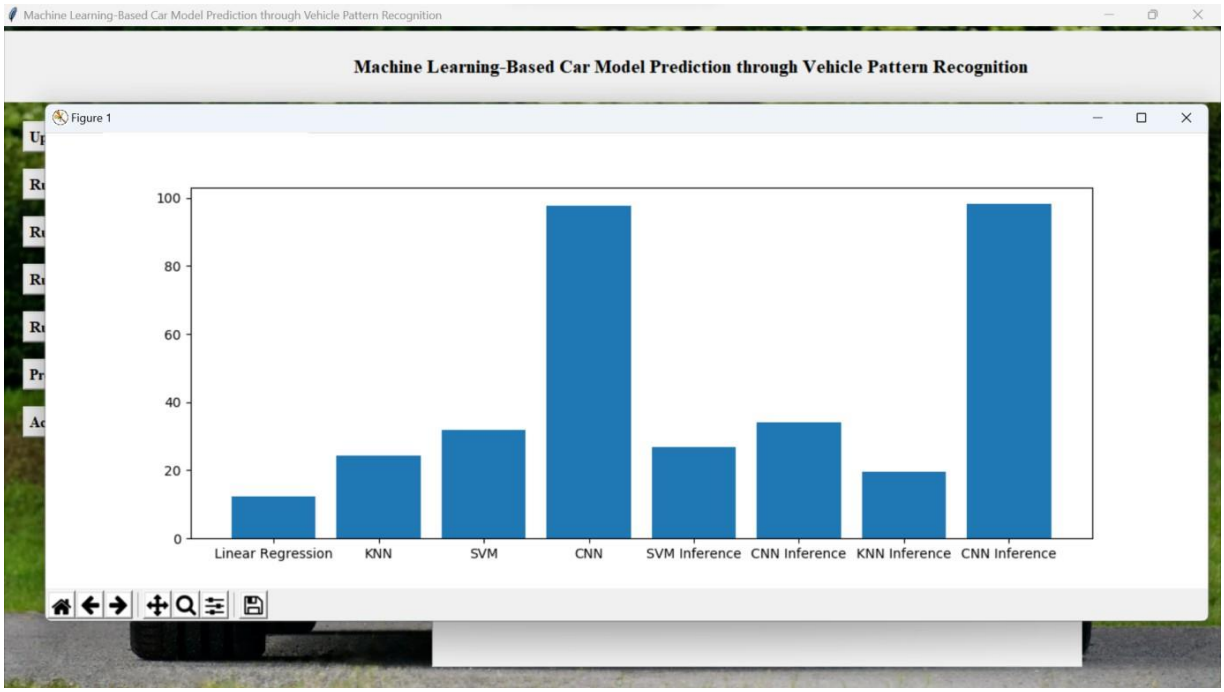


In above screen selecting and uploading 'bb2.jpg' and then click on 'Open' button to get below prediction result



Uploaded Prediction Model

Similarly, you can upload any image and get it model predicted. Now click on 'Accuracy Comparison Graph' button to get below screen



Algorithm & Accuracy Graph

In above screen x-axis represents algorithm name and y-axis represents accuracy of that algorithm In this project we are using know inference features and pattern features to predict car model using various machine learning algorithms such as SVM, KNN, CNN and Linear Regression. To implement this project, we have used Stanford car dataset and below screen shots showing some images from dataset

CHAPTER-9: CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

By employing frame subtraction and masking techniques, moving vehicles are segmented out. Speed is calculated using the time taken between frames and corner detected object traversed in that frames. Finally frame masking is used to differentiate between one or more vehicles. With an average error of +/-2 km/h speed detection was achieved

FUTURE ENHANCEMENT

Recognition can focus on improving feature extraction, model accuracy, real-time performance, and system scalability. Advanced feature extraction techniques, such as incorporating 3D modelling, LiDAR, and multi-sensor data, can enable the system to recognize cars more accurately from diverse perspectives. Leveraging cutting-edge machine learning models, including Vision Transformers and multi-modal architectures, can enhance prediction accuracy by combining visual data with other inputs like sound or textual information. Real-time performance can be improved through the use of lightweight edge AI models and distributed cloud-edge computing frameworks. Expanding the training dataset to include diverse car models, regions, and environmental conditions will ensure scalability and adaptability to global markets. Integration with smart systems, such as traffic management or IoT-enabled infrastructure, can enable practical applications like optimizing traffic flow and enhancing security. Furthermore, addressing ethical concerns through data anonymization, bias reduction, and adversarial robustness will ensure the system remains reliable and compliant with privacy regulations. These advancements, coupled with augmented reality interfaces and robust testing through synthetic data and simulations, can position the system as a powerful tool for industries like transportation, security, and automotive services**.**

CHAPTER 10: BIBLIOGRAPHY

Hannah Bast, Mattias Hertel, Mostafa M. Mohammed "Tokenization repair in the presence of spelling errors." Publisher: Association for Computational Linguistics, Volume: Proceedings of 25th conference on computational natural language learning, pages: 279-289, Issue: November 2021.

1. Yifei Hu, Xiaonan Jing, Youlim Ko "Misspelling Correction with Pre-trained Contextual Language Model", arXiv: 2101.03204v1 [cs.CL] 8 Jan 2021.
2. Xiangci Li, Hairong liu, Liang Huang "Context aware stand-alone Neural spelling correction." Findings of the Association for Computational Linguistics: EMNLP 2020, pages 407–414 November 16 - 20, 2020.
3. Ahmed Yunus, Md Masum **"**A context Free Spell Correction Method using Supervised Machine learning Algorithms", International Journal of Computer Applications, ISSN: 0975-8887, Volume 176, Pg No. 27, and June 2020.
4. Daniel Hladek , Jan Stas and Matus Pleva "Survey of Automatic Spelling Correction", Issue Human Computer Interaction for Intelligent Systems, Published: 13 October 2020.
5. Pravallika Etoori, Manoj Chinnakotla, Radhika Mamidi "Automatic Spelling Correction for Resource-Scare Languages using Deep Learning" Proceedings of ACL 2018, Student Research Workshop, Issue: July 15 - 20, 2018.
6. Christanti, M.V.; Naga, D.S., "Fast and accurate spelling correction using trie and Damerau-levenshtein distance bigram", Telkomnika (Telecommun. Comput. Electron. Control.) 2018, 16, 827–833.
7. S. M. El Atawy, A. Abd ElGhany *"*Automatic Spelling Correction based on n-Gram Model" International Journal of Computer Application, Volume: 182, Number: 11, Issue: 2018.
8. Tao Ge, Furu Wei, Ming Zhou "Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study", Cornell University article, Submitted on 3 Jul 2018, Version v5.
9. Keisuke sakaguchi, Kevin Duh, Matt post, Benjamin Van Durme "Robust word recognition via semi-character recurrent neural network." Association for the Advancement of Artificial Intelligence, volume: 2, issue: & Feb 2017.

10. Miikka Silfverberg, Pekka Kaupppinen, Krister Linden “Data-Driven Spelling Correction using Weighted Finite-State Methods” Proceedings of the ACL Workshop on Statistical NLP and Weighted Automata, Issue: August, 2016.
11. Daniel Hladek, Jan Stas, Jozef Juhar, “Learning string distance with smoothing for OCR spelling correction”, Multimedia Tools and Applications, Published: 07 December 2016.

YUKTHI INNOVENTION CERTIFICATE

MHRD'S INNOVATION CELL (GOVERNMENT OF INDIA)

INSTITUTION'S INNOVATION COUNCIL
MOE'S INNOVATION CELL

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

Institute Name:
Malla Reddy Institute of Technology & Science

Title of the Innovation/Prototype:
Machine Learning-Based Car Model Prediction through Vehicle Pattern Recognition

Team Lead Name:	Team Lead Email:	Team Lead Phone:	Team Lead Gender:
Rapelly Sahith	sahithrapelly2579@gmail.com	9553993878	Male

FY of Development:	Developed as part of:	Innovation Type:	TRL LEVEL:
2023-24	Academic Requirement/Study Project	Service	4

MRL Level:
MRL 1: Basic manufacturing implications identified

IRL Level:
IRL 1: Basic Research (Need Identification & Peer Review Publications) & Completed First-Pass Business Model Canvas (BMC)

Theme:
Defence & Security,

Define the problem and its relevance to today's market / sociaty / industry need:
Machine learning addresses the need for efficient and scalable decision-making based on data, as manual processes are often insufficient. Its relevance spans numerous sectors, such as automotive, security, and smart cities, where automated and accurate recognition systems are essential. ML enables the automation of complex tasks like vehicle recognition, which enhances operational efficiency, reduces human error, and optimizes resource allocation. The ability to handle vast amounts of data in real-time makes ML indispensable for modern technological and industrial applications.

Describe the Solution / Proposed / Developed:
The proposed solution involves developing a machine learning-based system for car model prediction through vehicle pattern recognition. This system utilizes advanced algorithms to analyze vehicle images, identifying and classifying car models accurately. The process includes preprocessing the images, extracting relevant features, and employing machine learning techniques to predict the car model. This solution enhances the accuracy and efficiency of vehicle identification, which is crucial for various applications such as traffic management, security, and smart city infrastructure.

Explain the uniqueness and distinctive features of the (product / process / service) solution:
The proposed solution stands out due to its use of advanced machine learning algorithms, such as convolutional neural networks (CNNs), which provide high accuracy in vehicle make and model identification. This solution is distinctive for its automation capabilities, reducing manual effort and speeding up operations. It is scalable, handling large volumes of data efficiently, and adaptable to diverse environments like urban areas and transportation hubs. Its real-time processing and ability to integrate with existing systems make it a versatile and robust tool for applications ranging from automated toll collection to security surveillanc.

How your proposed / developed (product / process / service) solution is different from similiar kind of product by the competitors if any:
The proposed machine learning-based car model prediction system differentiates itself from competitors by leveraging advanced deep learning models, such as Convolutional Neural Networks (CNNs), which provide superior accuracy in vehicle recognition. Unlike competitors, this solution offers scalability to handle large datasets and adaptability to diverse environmental conditions. Additionally, it integrates seamlessly with existing systems, enhancing operational efficiency across various applications, from traffic management to security surveillance. The system's high accuracy, real-time processing capability, and robust performance under varying conditions make it a unique and valuable tool in the market.

Is there any IP or Patentable Component associated with the Solution?:
No

Has the Solution Received any Innovation Grant/Seefund Support?:
No

Are there any Recognitions (National/International) Obtained by the Solution?:
No

***Is the Solution Commercialized either through Technology Transfer or Enterprise Development/Startup?:**
No

Had the Solution Received any Pre-Incubation/Incubation Support?:
No

Video URL:
https://docs.google.com/presentation/d/1Yjo01HwynQTW8ueU1qoe_LxVLhIYyVjZ/edit#slide=id.p1

Innovation Photograph:
View File

Downloaded on: 20-11-2024

This report is electronically generated against Yukti - National Innovation Repository Portal.