

# **ABSTRACT**

An approach to driving decision Strategy (DDS) A concept for driving decision- making for an autonomous car is based on machine learning, and it uses internal vehicles data, like Steering, RPM level, to forecast different types of behaviors, like speed (steering), changing lanes, etc. All methods at the time were designed to focus on exterior data, such as the state of the roads and the number of people, but not on internal variables. So, this strategy analyses the internal data to determine the steering state and lane changes effectively. All internal data will be gathered from sensors, saved on the cloud , read by the application , and then subjected to ML algorithms to ascertain steering angle or changing lanes. The DDS algorithm is based on a genetic algorithm to select the ideal gene values that aid in making better decisions and predictions and is used to implement this. Performance of the proposed DDS with genetic algorithm is compared to that of currently used ML techniques like Random Forest and MLP (multilayer perceptron algorithm). Compared to RF and MLP, the proposed DDS displays higher prediction accuracy.

## **LIST OF FIGURES**

<b>S.NO</b>	<b>Figure No</b>	<b>Description</b>	<b>Page No</b>
1	3.1	System Architecture of DDS	11
2	4.1	UML Diagram for DDS	18
3	4.2	Class Diagram for DDS	19
4	4.3	Sequence Diagram for DDS	19
5	4.4	Collaboration Diagram for DDS	20
6	5.1	Download Screen 1	36
7	5.2	Download Screen 2	37
8	5.3	Download Screen 3	37
9	5.4	Download Screen 4	38
10	5.5	Download Screen 5	38
11	5.6	Download Screen 6	39
12	5.7	Download Screen 7	39
13	5.8	Download Screen 8	

			40
14	5.9	Download Screen 9	40
15	5.10	Download Screen 10	41
16	5.11	Download Screen 11	41
17	5.21	Output Screen 1	42
18	5.22	Output Screen 2	42
19	5.23	Output Screen 3	43
20	5.24	Output Screen 4	43
21	5.25	Output Screen 5	44
22	5.26	Output Screen 6	44
23	5.27	Output Screen 7	45
24	5.28	Output Screen 8	45
25	5.29	Output Screen 9	46
26	5.3.0	Output Screen 10	46
27	5.3.1	Output Screen 11	47
28	5.3.2	Output Screen 12	47

## LIST OF TABLES

S.NO	TABLE NO	TABLE NAME	PAGE NO
1	2	Literature Survey of Existing models	4

# TABLE OF CONTENTS

<b>ABSTRACT</b>		<b>i</b>
<b>LIST OF FIGURES</b>		<b>ii</b>
<b>LIST OF TABLES</b>		<b>iv</b>
<b>SL.NO</b>	<b>CONTENT</b>	<b>PAGE NO</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 MOTIVATION	1
	1.2 PROBLEM DEFINITION	1
	1.3 OBJECTIVE OF THE PROJECT	1
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>3</b>
	2.1 INTRODUCTION	5
	2.2 EXISTING SYSTEM	5
	2.3 DISADVANTAGES OF EXISTING SYSTEM	6
	2.4 PROPOSED SYSTEM	7
	2.5 CONCLUSION	8
<b>3</b>	<b>ANALYSIS</b>	<b>9</b>
	3.1 INTRODUCTION	9
	3.1.1 SOFTWARE REQUIREMENT	10
	3.1.2 HARDWARE REQUIREMENTS	10
	3.2 SYSTEM ARCHITECTURE	11
	3.3 SYSTEM STUDY FEASIBILITY STUDY	11
	3.4 SOFTWARE REQUIREMENT SPECIFICATION	12
	3.5 OBJECTIVES	14
	3.6 ALGORITHMS AND FLOWCHARTS	15
	3.7 CONCLUSION	16

<b>4</b>	<b>DESIGN</b>	<b>17</b>
	4.1 UML DIAGRAM	18
	4.2 USE CASE DIAGRAM	19
	4.3 CLASS DIAGRAM	20
	4.4 SEQUENCE DIAGRAM	20
	4.5 COLLABORATION DIAGRAM	21
	4.6 MODULE	21
	4.7 DESIGN AND ORGANIZATION	22
	4.8 CONCLUSION	23
<b>5</b>	<b>IMPLEMENTATION AND RESULTS</b>	<b>24</b>
	5.1 SOFTWARE ENVIRONMENT	24
	5.2 OUTPUT SCREENS	44
	5.3 CODES	50
<b>6</b>	<b>TESTING AND VALIDATION</b>	<b>56</b>
<b>7</b>	<b>CONCLUSION</b>	<b>59</b>
<b>8</b>	<b>REFERENCES</b>	<b>60</b>

# **1. INTRODUCTION**

## **1.1 MOTIVATION**

Companies all over the world are actively developing technologies for sophisticated autonomous vehicles, which are currently in the fourth stage of development. Self-driving cars' operational framework can be divided into three key levels: recognition, judgment, and control. Vehicles are outfitted with a variety of sensors during the recognition phase, including GPS, cameras, and radar. The judgment phase develops a driving strategy based on the data collected. Once the driving environment has been identified, it is analyzed, which leads to the development of appropriate driving plans and objectives. Following the completion of the control step, the vehicle begins driving autonomously. An autonomous vehicle executes a series of actions to reach its destination, autonomously iterating through the recognition, judgment, and control steps.

## **1.2 PROBLEM DEFINITION**

As their capabilities improve, an increase in the number of sensors poses the risk of overloading the vehicle's electrical system. In self-driving cars, in-vehicle computers process data collected by these sensors. The speed of judgment and control slows as the amount of computed data increases, potentially jeopardizing the vehicle's stability. Sensors such as GPS, cameras, and radar gather information about the vehicle's surroundings during the recognition phase. The proliferation of sensors, on the other hand, may result in vehicle overload for self-driving cars. Onboard computers are in charge of processing sensor data, and as the volume of data increases, it can impede decision-making and traffic management, putting the vehicle's stability at risk.

## **1.3 OBJECTIVE OF THE PROJECT**

The Driving Decision Strategy (DDS) for autonomous vehicles is built with a multifaceted goal in mind: to improve the overall functionality and safety of these self-driving systems. The DDS's core goal is to revolutionize decision-making processes within the context of autonomous vehicles. The strategy aims to optimize real-time responses to dynamic driving conditions by leveraging the power of machine learning. One critical goal is to reduce in-vehicle data processing requirements, ensuring that decision-making remains quick and efficient.

The DDS hopes to store and leverage historical driving data through the integration of cloud computing, fostering adaptive learning and enhancing the vehicle's ability to navigate through various scenarios.

The "driving decision strategy for an autonomous vehicle" system is a complex software solution that allows a vehicle to make autonomous driving decisions using machine learning algorithms. The system is intended to collect data from various sources, such as sensors and cameras, to process that data in real-time using machine learning techniques, and to make driving decisions based on the processed data.

Several components, such as data collection and pre-processing modules, a machine learning model, and a decision-making module, are typically included in the system. The data collection and pre- processing modules collect data from the vehicle's various sensors and cameras and pre-process it to make it suitable for the machine learning model.



## 2. LITERATURE SURVEY

Y. N. Jeong, S. R. Son, E.H. Jeong and B. K. Lee, “An Integrated Self- Diagnosis System for an Autonomous Vehicle Based on an IoT Gateway and Deep Learning” *Applied Sciences*, vol. 8, no. 7, July 2018.

This paper proposes "An Integrated Self-diagnosis System (ISS) for an Autonomous Vehicle based totally on an Internet of Things (IoT) Gateway and Deep Learning," which collects records from an independent vehicle's sensors, diagnoses itself and the have an impact on between its components the usage of Deep Learning, and notifies the driver of the results. Three modules make up the ISS. The first In-Car Gateway Module (In-VGM) takes facts from in-vehicle sensors, such as media records from a black box, riding radar, and car manipulate messages, and sends every piece of statistics over every Controller Area Network (CAN) to the on-board diagnostics (OBD) or actuators with the aid of the, Flex Ray, and Media Oriented Systems Transport (MOST) protocols.

The statistics from in-vehicle sensors is dispatched to the CAN or Flex Ray protocol, whilst media facts acquired whilst using is dispatched to the MOST protocol. A vacation spot protocol message kind is created from various kinds of messages that have been transferred. The 2nd Optimized Deep Learning Module (ODLM) generates the Training Dataset the usage of information obtained from in-car sensors and calculates the chance of car components and consumables, as nicely as the threat of different components influenced through a faulty part. to enhance the self-diagnosis velocity and decrease the device overhead, whilst a V2X primarily based Accident Notification Service (VANS) informs the adjoining motors and infrastructures of the self- diagnosis result analyzed with the aid of the OBD. This paper improves upon the simultaneous message transmission effectivity via the In-VGM by way of 15.25% and diminishes the mastering error price of a Neural Network algorithm via the ODLM through about 5.5%. The following table consists of referred survey papers and describes techniques and details

S.No	Title	Link	Authors	Techniques used
1	An Integrated Self Diagnosis System for an Autonomous Vehicle andBased on an IoT Gateway and Deep Learning	<a href="https://www.mdpi.com/2076-3417/8/7/1164">https://www.mdpi.com/2076-3417/8/7/1164</a>	Y. N.Jeong, S. R.Son, E.H. Jeong and B. K. Lee	Deep Learning
2	Discrete plane segmentation and estimation from a point cloud using local geometric patterns	<a href="https://www.researchgate.net/">https://www.researchgate.net/</a>	Yukiko Kenmochi, Lilian Buzer, Akihiro Sugimoto, Ikuko Shimizu	Automation and Computing
3	Vehicle trajectory prediction based on Hidden Markov Model	<a href="https://itiis.org/digital-library/21157">https://itiis.org/digital-library/21157</a>	NingYe, Yingya Zhang, Ruchuan Wang, Reza Malekian	The KSII Transactions on Internet and Information Systems

Table 2: Literature Survey of Existing models

## **2.1 INTRODUCTION**

Companies all over the world are actively developing technologies for sophisticated autonomous vehicles, which are currently in the fourth stage of development. Self-driving cars' operational framework can be divided into three key levels: recognition, judgment, and control. Vehicles are outfitted with a variety of sensors during the recognition phase, including GPS, cameras, and radar. The recognition step involves recognizing and collecting information about the environment using various sensors in vehicles such as GPS, camera, and radar. Based on the recognized information, the judgment step determines the driving strategy.

Then, this step identifies and analyzes the conditions under which the vehicle is operating, and it determines the driving plans that are appropriate for the driving environment and the objectives. The control step determines the vehicle's speed, direction, and so on, and the vehicle begins driving on its own. To arrive at its destination, an autonomous driving vehicle performs a variety of actions, repeating the steps of recognition, judgment, and control on its own.

However, as the performance of self-driving cars improves, so does the number of sensors used to recognize data. Increases in these sensors can result in in-vehicle overload. In-vehicle computers are used by self-driving cars to compute data collected by sensors. Overload can affect the speed of judgment and control as the amount of computed data increases. These issues may jeopardize the vehicle's stability. Some studies have developed hardware that can perform deep running operations inside the vehicle to prevent overload, while others use the cloud to compute the vehicle's sensor data. Existing studies, on the other hand, only use real-time data, such as images and sensor data currently collected from vehicles, to determine how the vehicle is driving.

## **2.2 EXISTING SYSTEM**

Companies around the world are developing technologies for advanced autonomous vehicles, which are currently in the 4th stage of development. The principle of operation of self-driving cars can be classified into three levels: recognition, judgement, and control. As part of the recognition process, vehicles are equipped with various sensors, including GPS, cameras, and radar. As a result of this information, the judgement step determines a driving strategy. When the driving environment is identified, it is analyzed and appropriate driving plans are developed and the objectives. Vehicle starts driving on its own after the control step has been completed. To reach its destination, an autonomous vehicle performs a series of actions, repeating on its own the steps of recognition, judgement, and control. Global corporations are actively

developing advanced technologies for autonomous vehicles, which are currently in the fourth stage of their evolutionary journey. The self-driving car's operational framework is divided into three distinct levels: recognition, judgment, and control. Vehicles are outfitted with a variety of sensors during the recognition phase, including GPS, cameras, and radar. The data collected by these sensors is crucial in the subsequent decision step, which determines a driving strategy. Following the identification of the driving environment, a thorough analysis is carried out, which leads to the development of appropriate driving plans and objectives. Following the completion of the control step, the vehicle begins driving autonomously.

## **2.3 DISADVANTAGES OF EXISTING SYSTEM**

**Sensor Dependence:** The reliance on various sensors, such as GPS, cameras, and radar, poses a vulnerability. External factors such as inclement weather, sensor malfunctions, or physical obstructions can impede data collection accuracy, possibly compromising the DDS's overall effectiveness.

**Processing Overhead:** On-board processing of large amounts of sensor data may result in computational overhead. As the volume of data increases, there is a risk that the decision-making process will be slowed during the judgment and control phases. This delay could have an impact on the autonomous vehicle's real-time responsiveness.

**Lack of Redundancy:** The DDS may fail due to a sensor malfunction or a breakdown in the decision-making process if there is no robust redundancy system in place. Without adequate backup mechanisms, the autonomous vehicle's dependability and safety may be jeopardized.

**Data Security and Privacy Concerns:** Storing driving data in the cloud raises data security and privacy concerns. The reliance on third-party cloud computing services introduces potential vulnerabilities, and ensuring the security of sensitive driving data becomes critical to preventing unauthorized access or misuse.

**Limited Learning from Historical Data:** As the system repeats the steps of recognition, judgment, and control, the system's ability to learn from historical data may be limited. A more advanced learning mechanism could improve the system's ability to adapt and improve over time.

## 2.4 PROPOSED SYSTEM

The current methodologies in the field of autonomous vehicle technology have primarily focused on external factors such as road conditions and pedestrian presence, inadvertently ignoring the exploration of critical internal parameters. In a break from the norm, the author of this approach acknowledges the importance of internal values in the efficient determination of steering conditions and lane changes. Instead of relying solely on external stimuli, this novel approach delves into the complexities of internal data. The method entails the systematic collection of internal data via sensors embedded in the vehicle. These sensors record a wide range of information about the vehicle's internal state, including engine performance, tire conditions, and internal temperature. The accumulated internal data is then transmitted and securely stored in the cloud, resulting in the creation of a centralized repository. In the following phase, an application interacts with the cloud- stored data to extract useful insights. This is a departure from traditional systems, which rely heavily on real-time, external data. The application's use of machine learning algorithms, a key enabler for making predictions and determinations about the vehicle's steering condition and potential lane changes, is transformative.

The machine learning algorithms, powered by the extensive internal dataset, enable more nuanced and adaptive decision-making. The system gains the ability to anticipate and respond to the vehicle's internal dynamics by assimilating historical patterns and real-time internal data. This improves not only the precision of steering decisions, but also the timing and strategy for lane changes.

This innovative approach, in essence, represents a paradigm shift by recognizing the critical role of internal data in autonomous vehicle decision-making. The system demonstrates a holistic understanding of the vehicle's internal state by leveraging cloud storage and machine learning algorithms, ultimately contributing to more efficient, adaptive, and safer driving experiences.

## ADVANTAGES OF PROPOSED SYSTEM

**Predictive Capability Enhancement:** Internal values provide a more comprehensive understanding of the vehicle's condition, including engine performance, tire health, and internal temperature. This wealth of data enables machine learning algorithms to more accurately predict steering conditions and lane changes. The system gains heightened predictive capability by considering both historical and real-time internal data, resulting in more precise and timely decisions.

**Improved Vehicle Dynamics Adaptability:** Internal data analysis allows the system to adapt to the vehicle's specific dynamics. Internal factors that influence steering conditions and lane changes are frequently overlooked in traditional methods. By focusing on internal values and utilizing machine learning, the system becomes more adaptable, recognizing and responding to the vehicle's distinct characteristics and performance nuances. This adaptability helps to make driving safer and more efficient.

**Maintenance Strategies That Work:** Internal data collection and analysis provide valuable insights into the health and performance of various vehicle components. The system can recommend proactive maintenance interventions by identifying patterns and anomalies in this data. This contributes to optimized maintenance strategies, lowering the risk of unexpected breakdowns and improving overall vehicle reliability and longevity.

## 2.5 CONCLUSION

Finally, the new method of shifting focus from sole reliance on external data to a more holistic consideration of internal values represents a significant advancement in the field of autonomous vehicle technology. The system achieves heightened predictive capability and adaptability to the unique dynamics of each vehicle by meticulously analysing internal data encompassing the intricate details of the vehicle's condition. This not only leads to more precise and timely determinations of steering conditions and lane changes, but it also contributes to optimized maintenance strategies, thereby increasing overall reliability. Furthermore, by reducing reliance on real-time external data and leveraging cloud storage for internal information, the proposed method improves the system's autonomy and robustness, allowing it to navigate a wide range of driving conditions.

## **3.ANALYSIS**

### **3.1 INTRODUCTION**

System analysis is a process for reviewing a technological system for troubleshooting, development, or improvement. Such a system might be a software implementation, like a system or application program. It is important to consider system analysis as one phase in a larger process, the systems development life cycle (SDLC). In the system analysis phase, analysts are concerned with outlining a proposed solution to a defined problem. In doing so, they consider how viable and effective the product is or will be. System analysts consider the system's overarching goals, which they can then break down into components or modules to enable individual analyses. System analysis is the first of these activities and forms the foundation for the rest of the software development process. System analysis outlines a proposed solution to a defined problem. Analysis means "to take apart" (Boehm, 2003), and so system analysis involves breaking down a system to identify its functions, roles, and the environment in which it will be expected to operate. Within this activity are the tasks of performing a feasibility study and requirements engineering. System analysis is therefore very important throughout any software development process as it plays a significant determining factor in the success of any software project in terms of usefulness and delivery within established constraints; and based on how well it is performed, it can make the difference between software products which are maintainable and those which quickly become obsolete. Whether software is developed sequentially or iteratively, system analysis is performed in some way.

The output of system analysis is a system specification document which becomes a major input into other software development process activities such as system design, programming, and testing. The specification document outlines in detail what the software product should do and often includes a project plan, software models, prototyping results, formal specifications, and verification test data. This can lead to disastrous outcomes, including customer dissatisfaction due to contextually useless software being delivered. Other negative effects occur when the inaccuracy is discovered, and include an increase in time, cost, and effort to produce correct software. System analysis uncovers what software does and how it should behave. This is important information when maintaining software to make corrections or add functionality. Software evolution includes modifications to software but should not compromise its functionality. Throughout software development, it is likely that those who build the software

product will not be the ones to maintain it. It is therefore of paramount importance that a guideline exists for reference, so that the software evolves within the scope of its intended functionality. Without thorough system analysis, there is an increased risk of software becoming obsolete due to an inability to properly maintain the software, caused by a lack of understanding of the system specification. In conclusion, system analysis is the first step in any software development process, whether sequential or iterative.

It involves defining what software should do to be useful in solving an identified problem and its tasks result in a system specification document which feeds into subsequent software development process activities. The strategic position of system analysis in the software development process means it can largely affect the success of a software project in terms of usefulness, cost, schedule, and maintenance. If system analysis is done incorrectly, other process activities will suffer, which ultimately leads to customer dissatisfaction and extensive rework. After software is deployed, the system specification document becomes a guide for software maintenance without changing the system's functionality. When system analysis is completed properly, the software project has the best chance of being undertaken successfully and within given constraints.

### **3.1.1 Software Requirement**

Software requirements address the definition of software application requirements and pre-requisites that require computer installation to provide System performance. These prerequisites or requirements are not usually included in the software installation package and need to be installed separately before the software can be installed.

- Operating system : Windows8 or Above.
- Coding Language : python

### **3.1.2 Hardware Requirement**

The most common set of requirements defined by any application or software application for virtual computer applications, also known as hardware, Hardware Requirements list is usually accompanied by a hardware compliance list (HCL), especially if there are applications

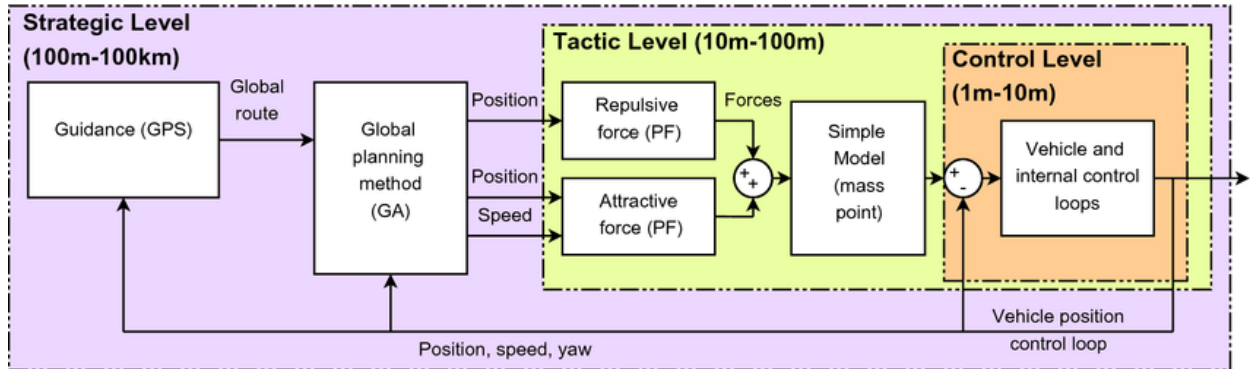
The HCL list checks hardware devices that are tested, compatible, and sometimes not compatible with a specific application or application. The following sections discuss various aspects of hardware requirements.

- System : i3 or above.



- Ram : 4 GB.
- Hard Disk : 40 GB

## 3.2 SYSTEM ARCHITECTURE



**fig 3.1 System Architecture of DDS**

## 3.3 SYSTEM STUDY 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.

Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

### **Economic 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.

### **3.4 SOFTWARE REQUIREMENT SPECIFICATION**

The reason for this SRS record is to distinguish the necessities and functionalities for Intelligent Network Backup Tool. The SRS will characterize how our group and the customer consider the last item and the attributes or usefulness it must have. This record additionally makes a note of the discretionary prerequisites which we intend to execute yet are not required for the working of the venture. This stage assesses the required necessities for the Images Processing for an orderly method for assessing the prerequisites a few procedures are included. The initial step associated with dissecting the prerequisites of the framework is perceiving the idea of framework for a solid examination and all the case are defined to better comprehend the investigation of the dataset. Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management. Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs

rather than attempting to mold user expectations to fit the requirements. Requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people.

### **FUNCTIONAL REQUIREMENTS:**

Functional requirements should include functions performed by a specific screen outline workflows performed by the system and other business or compliance requirements the system must meet. Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

The functional specification describes what the system must do, how the system does it is described in the design specification. If a user requirement specification was written, all requirements outlined in the user requirements specifications should be addressed in the functional requirements.

### **NON-FUNCTIONAL REQUIREMENTS:**

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (i.e., how precise are the systems numerical answers.).

### **INPUT & 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 that it provides security and ease of use while retaining privacy. Input Design considered the following things:

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

### **3.5 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 the results of processing are communicated to the users and to other systems through outputs. In output design it is determined how the information is to be displaced for immediate need and the hard copy output. It is the most important and direct source of 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 documents, reports, or other formats that contain information produced by the system.

### **3.6 ALGORITHMS AND FLOWCHART**

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything.

The concept of deep learning is not new. It has been around for a couple of years now. It's hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

#### **A formal definition of deep learning is- neurons**

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In the human brain there are approximately 100 billion neurons altogether this is a picture of an individual neuron, and each neuron is connected through thousands of their neighbors. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

#### **SQL**

SQL (Structured Query Language) is typically used to interact with and manage the underlying databases that store various types of data relevant to the vehicle's decision-making process in a Driving Decision Strategy (DDS) based on Machine Learning for an autonomous vehicle. SQL is a critical tool for managing and accessing the underlying data required for decision-making. SQL is used to create and manage databases that contain critical information such as historical driving data, internal vehicle states, and environmental conditions. Queries are written to retrieve, update, and analyze data as quickly as possible. The language allows the DDS to access current vehicle parameters and historical data, assisting machine learning algorithms in predicting optimal driving strategies. SQL queries are also useful for generating insights from stored data, which contributes to the DDS's continuous improvement.

## **NEURAL NETWORKS:**

Convolutional Neural Networks (CNNs) :

CNNs are effective at processing image data, making them suitable for tasks such as recognizing objects in the vehicle's surroundings. Recurrent Neural Networks (RNNs): RNNs are useful for predicting actions based on sequential information because they can capture temporal dependencies in data.

## **3.7 CONCLUSION**

DDS executes the genetic algorithm based on accumulated data to determine the vehicle's optimal driving strategy according to the slope and curvature of the road in which the vehicle is driving and visualizes the driving and consumables conditions of an autonomous vehicle to provide drivers. To verify the validity of the DDS, experiments were conducted on the DDS to select an optimal driving strategy by analyzing data from an autonomous vehicle. Though the DDS has a similar accuracy to the MLP, it determines the optimal driving strategy 40% faster than it. And the DDS has a higher accuracy of 22% than RF and determines the optimal driving strategy 20% faster than it. Thus, the DDS is best suited for determining the optimal driving strategy that requires accuracy and real-time.

## 4. DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analysed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution to the problem specified by the requirement document.

This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phase, particularly testing and maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design, also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module are usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other words, in system design the attention is on what components are needed,

while in detailed design how the components can be implemented in software is the issue. Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified. During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user. Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. The organization code and title of the key points of contact (and alternates if appropriate) for the information system development effort. These points of contact should include the Project Manager, System Proponent, User Organization, Quality Assurance (QA) Manager, Security Manager, and Configuration Manager, as appropriate.

#### **4.1 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:

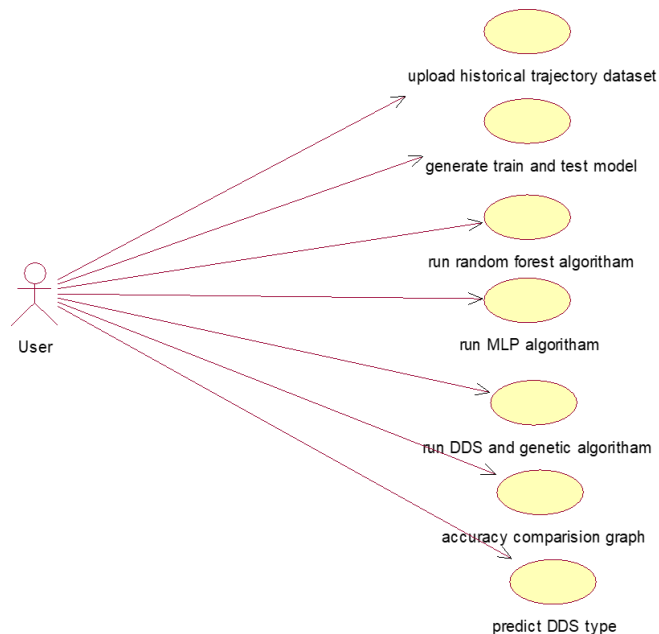
The Primary goals in the design of the UML are as follows:

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.
7. Integrate best practices.

## 4.2 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.

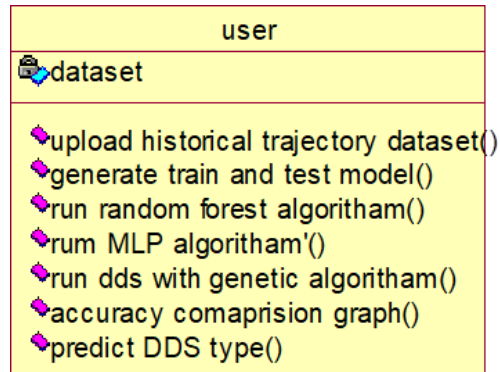
Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**fig 4.1: UML Diagram for DDS**

### 4.3 CLASS DIAGRAM

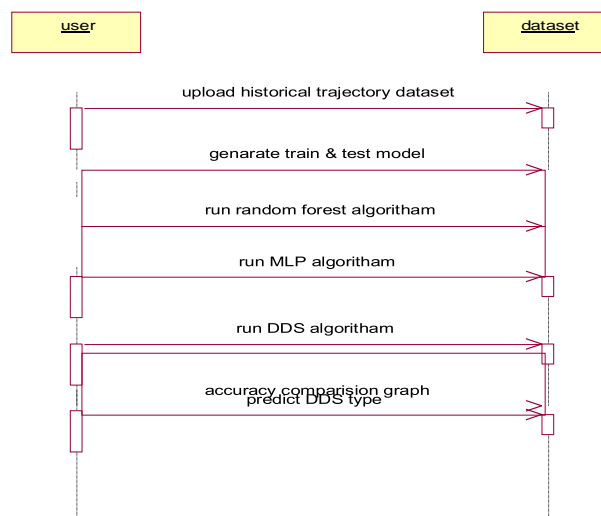
In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class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



**fig 4.2:Class Diagram for DDS**

### 4.4 SEQUENCE DIAGRAM

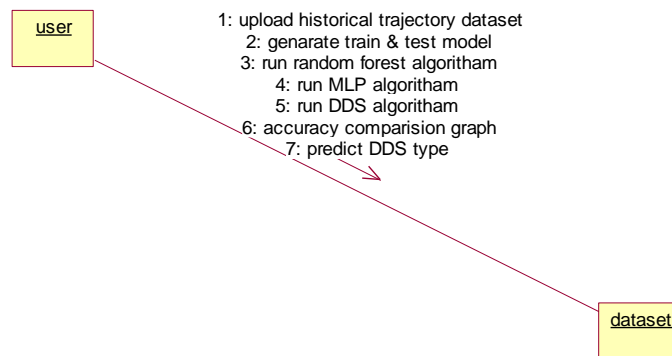
A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**fig 4.3 Sequence Diagram for DDS**

## 4.5 COLLABRATION 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 4.4 Collaboration Diagram for DDS**

## 4.6 MODULE

1. Upload Historical Trajectory Dataset: Upload Historical Trajectory Dataset' button and upload dataset.
2. Generate Train & Test Model: Generate Train & Test Model' button to read dataset and to split dataset into train and test part to generate machine learning train model
3. Run MLP Algorithm: Run MLP Algorithm' button to train MLP model and to calculate its accuracy.
4. Run DDS with Genetic Algorithm: Run DDS with Genetic Algorithm button to train DDS and to calculate its prediction accuracy.
5. Predict DDS Type: Predict DDS Type' button to predict test data.

## 4.7 DESIGN AND ORGANIZATION

### AN INPUT OF THE DDS

The DDS makes use of two types of data: training data sets used to train a genetic algorithm and post-training output data. Each training data set consists of a training input and a training output. The training input includes values representing the road condition as well as sensor messages received from the vehicle. In contrast, the driving information values are represented in the training output. The DDS normalizes the data to aid the genetic algorithm's learning process. Equation (1) formally expresses the training data sets used for algorithm learning.

$T = \{\{TX1, TY1\}, \{TX1, TY2\}, \dots, \{TX10000, TY1000\}\}$  (1) Here, TX means a training input and TY means a training output. To learn the genetic algorithm, 10000 of the past data are used and the values for all slopes and curvatures must be the same.

equation (2) represents a set of chromosomes in the genetic algorithm.

$Y = \{yr = RPM/10000, ya = Angle/100, ys = Speed/1000, ylc = \text{Whether the vehicle has changed lanes}\}$  (2) The DDS generates a set of initial 100 chromosomes normalized between 0 and 1.

### A LEARNING OF THE DDS

The genetic algorithm is taught to the DLS using training data sets and an initial set of chromosomes. First, the DDS sends the DDS 100 sets of initial chromosomes that are randomly assigned for learning RPM, speed, steering angle, and lane change.

The DDS generates 0.05% of the child genes at random as mutant genes, as well as a set of mutant chromosomes via reverse operations. The DDS chooses one of the child gene sets generated at random and then chooses one of the genes from that set at random. The selected gene value is then reversed using a fixed maximum value that the gene can have. As a result, the DDS generates 0.05 percent of the total set of mutant genes.

If the learning times are set to 1000 and a set of chromosomes with a suitability less than 0.001 is generated, the DDS terminates the genetic algorithm's learning and stores the speed, RPM, lane change, and steering angle for road slope and curvature.

### A USAGE OF THE DDS

The DDS determines an appropriate set of chromosomes for a specific slope and curvature of the driving road, generates an optimal driving table, and stores the set of chromosomes in the optimal driving table. Because the genetic algorithm computes the optimal driving strategy based on specific slopes and curvature by analyzing sufficient historical data, it does not require real-time deep- running, machine-running, and other operations.

The DDS searches for a set of chromosomes with similar slope and curvature values in the optimal driving table after inputting the slope and curvature of the vehicle which is currently in driving. If it exists, the DDS transmits it to the vehicle to implement the optimal driving strategy.

## **4.8 CONCLUSION**

The DDS determines the vehicle's optimal driving strategy at a faster rate than existing methods because it sends only the key data required to determine the vehicle's optimal driving strategy to the cloud and analyzes the data using the genetic algorithm. However, the DDS experiments were carried out in virtual environments using PCs, and there were insufficient resources for visualization. Future research should put the DDS to the test by applying it to actual vehicles, and professional designers should improve the completeness of visualization components.

## 5. IMPLEMENTATION & RESULTS

### 5.1 SOFTWARE ENVIRONMENT

#### **What is 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 must 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 library 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, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

#### **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. Adjustable**

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. This 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. 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 Carbon NELLE

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.

This was all about the Advantages and Disadvantages of Python Programming Language.

## **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 Venner<sup>1</sup>, 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 En 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 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.

### **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 Machine 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 This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer.

## **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 Geeks for Geeks.

## **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 color, 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 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.

## **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 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 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 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 IPython 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 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 :

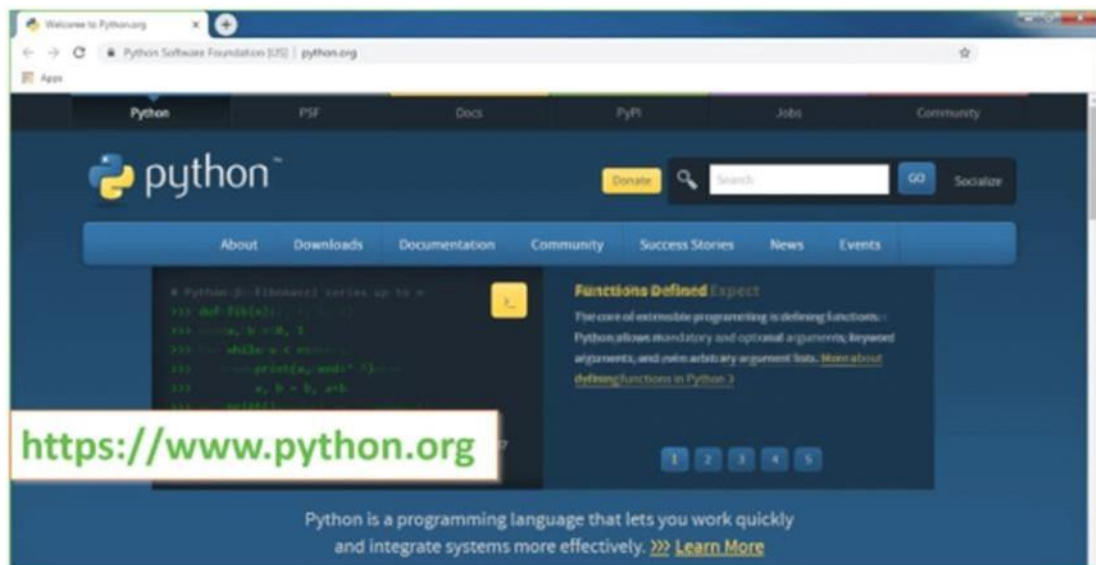
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other 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 Cheatsheet 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 5.1 :** Download Screen 1

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

**Step 2:** Click on the Download Tab.










**Fig 5.2:** Download Screen 2

**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

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>

**Fig 5.3:** Download Screen 3

**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.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPU
Unpacked source tarball	Source release		68111671a5b2db4a77b9d0137079be	23817663	36G
32 compressed source tarball	Source release		d33e4aa66097051c3eca45ee3604803	17131432	36G
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6a28b4f57583da71a44c7ba8ce88e6	34898416	36G
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	3dd607c38217a45773b95ea936b2a1f	28882845	36G
Windows help file	Windows		d63999573a2b58b2ac58acde8b477ed2	8131761	36G
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9809c3c7689e3b8a8e83184a48723a2	7504391	36G
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	a702b4b0a76d85dc35c3a583e563400	26882968	36G
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	28c31c5088bd73a8e53a3b6351b4bd2	1362904	36G
Windows x86 embeddable zip file	Windows		9fac18d18841879fda94123574129d8	6741626	36G
Windows x86 executable installer	Windows		33c1802942a5446a3d945147e394789	25663848	36G
Windows x86 web-based installer	Windows		1b670c1e5d317d82c30983ea371607c	1324606	36G

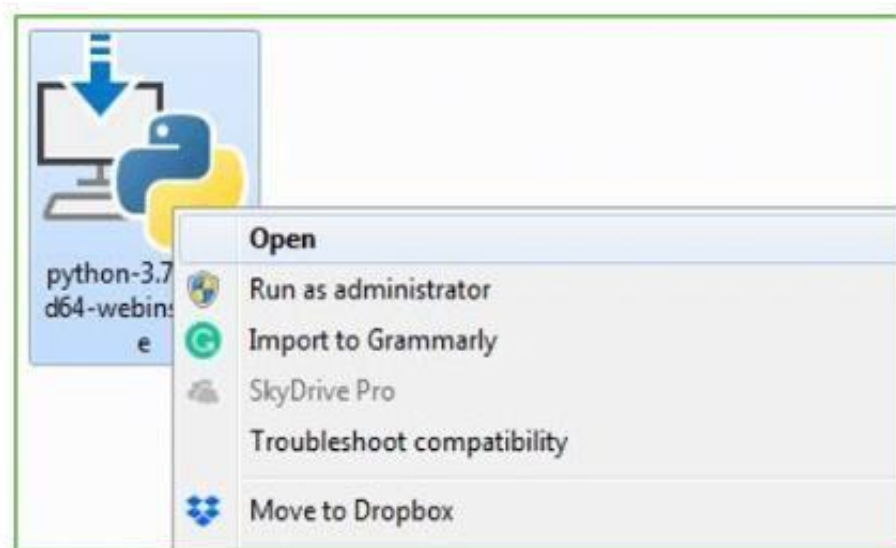
**Fig 5.4:** Download Screen 4

- 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.

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 5.5 :** Download Screen 5

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



**Fig 5.6 :** Download Screen 6

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



**Fig 5.7 :** Download Screen 7

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

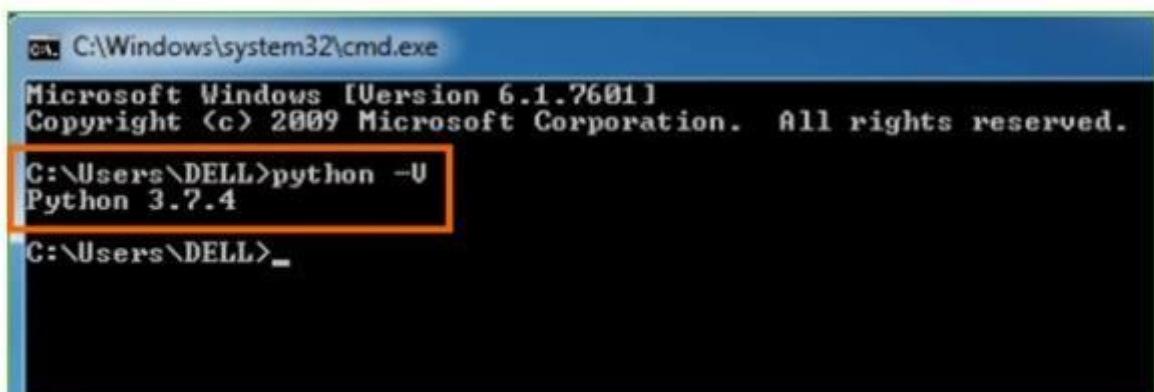
**Step 2:** In the Windows Run Command, type “cmd”.



**Fig 5.8 :** Download Screen 8

**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 5.9 :** Download Screen 9

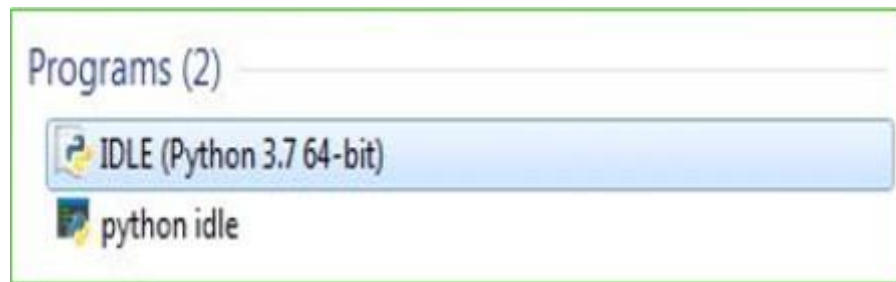
**Step 5:** You will get the answer as 3.7.4

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”.

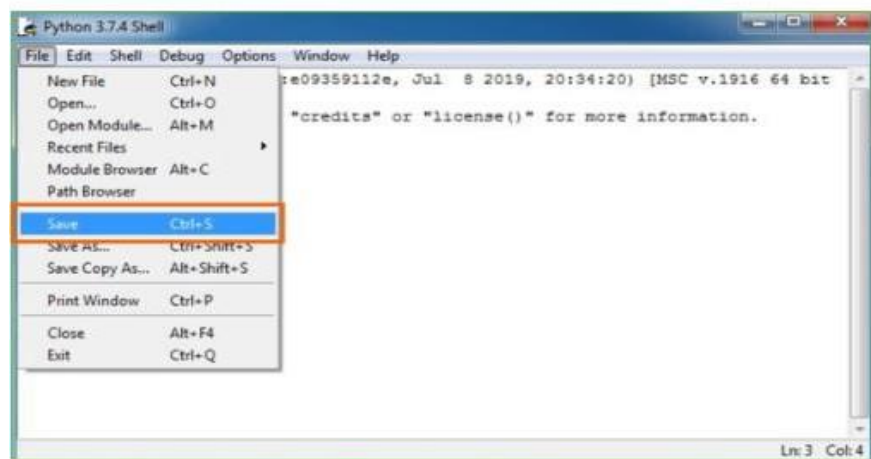




**Fig 5.10 :** Download Screen 10

**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



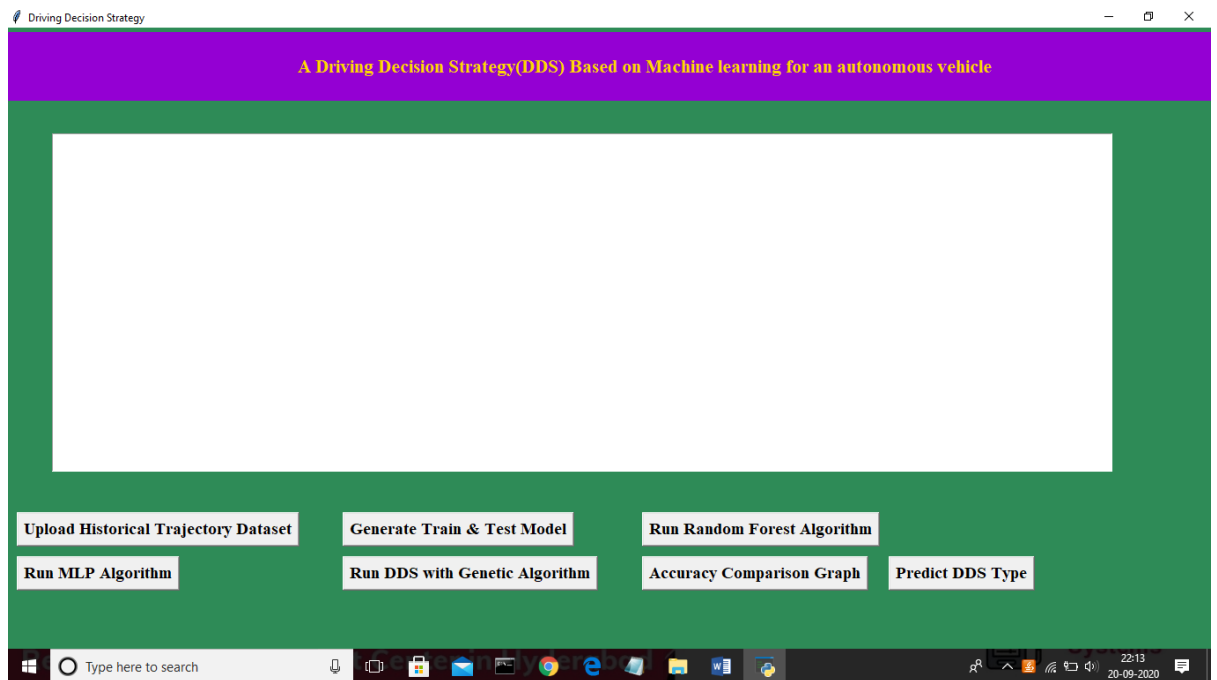
**Fig 5.11 :** Download Screen 11

**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

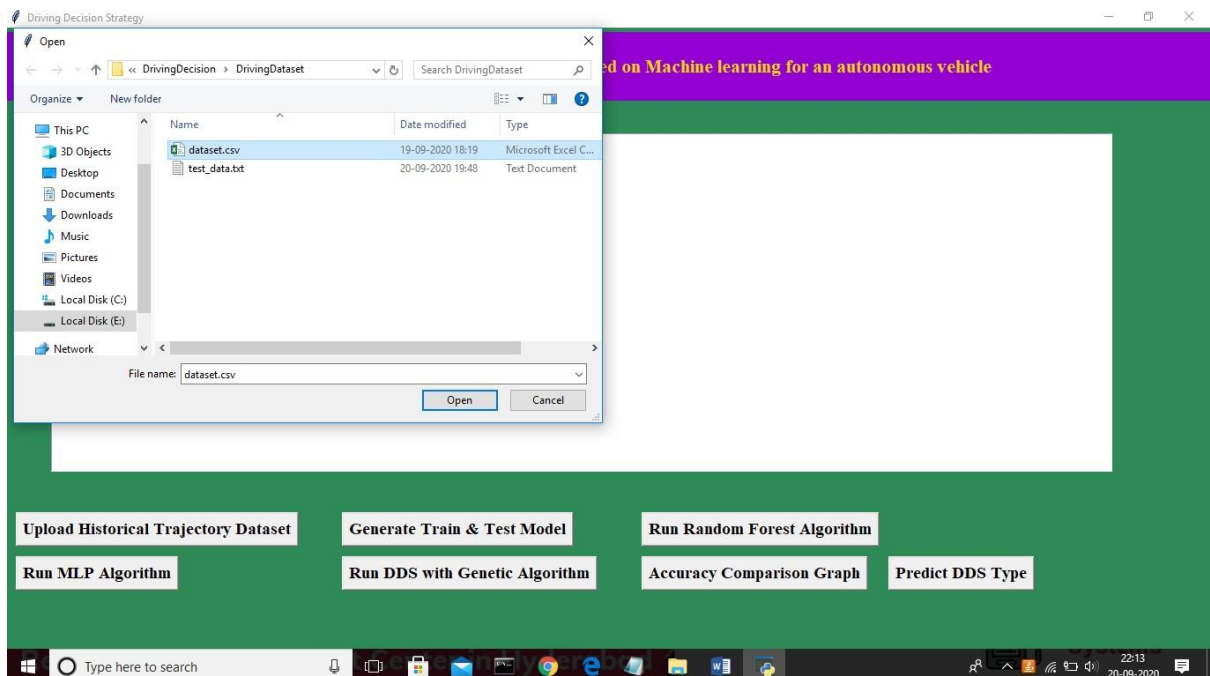
## 5.2 OUTPUT SCREENS

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



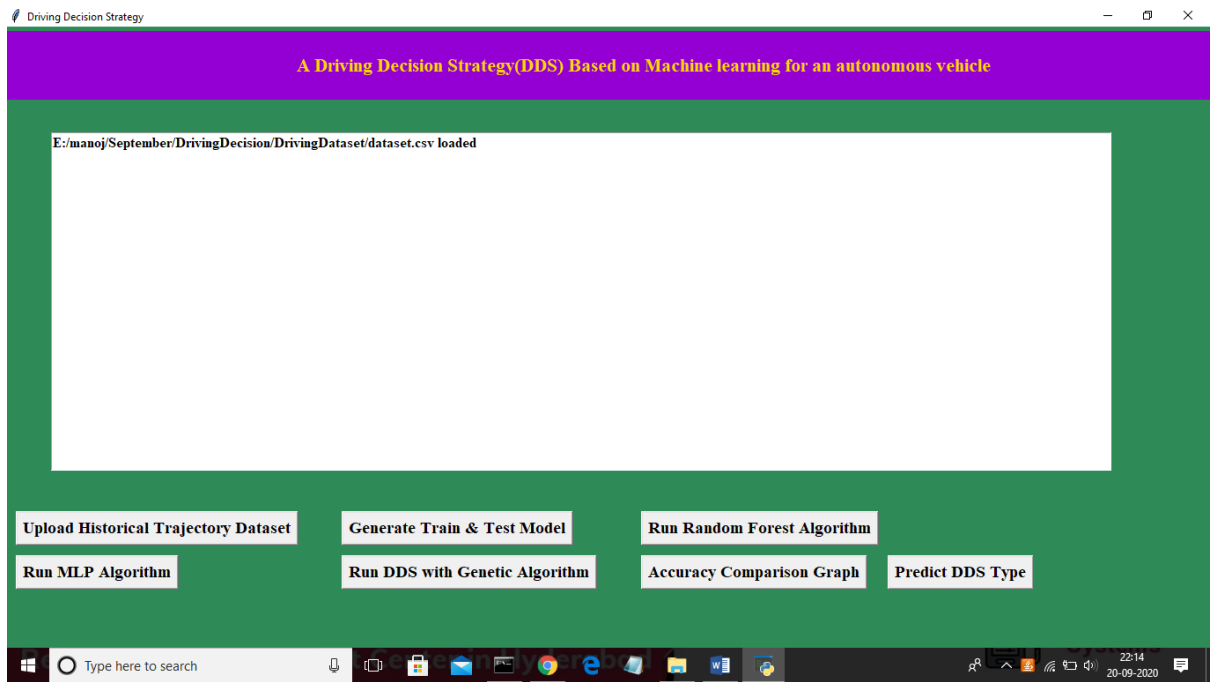
**Fig 5.21 : Output Screen 1**

In above screen click on 'Upload Historical Trajectory Dataset' button and upload dataset



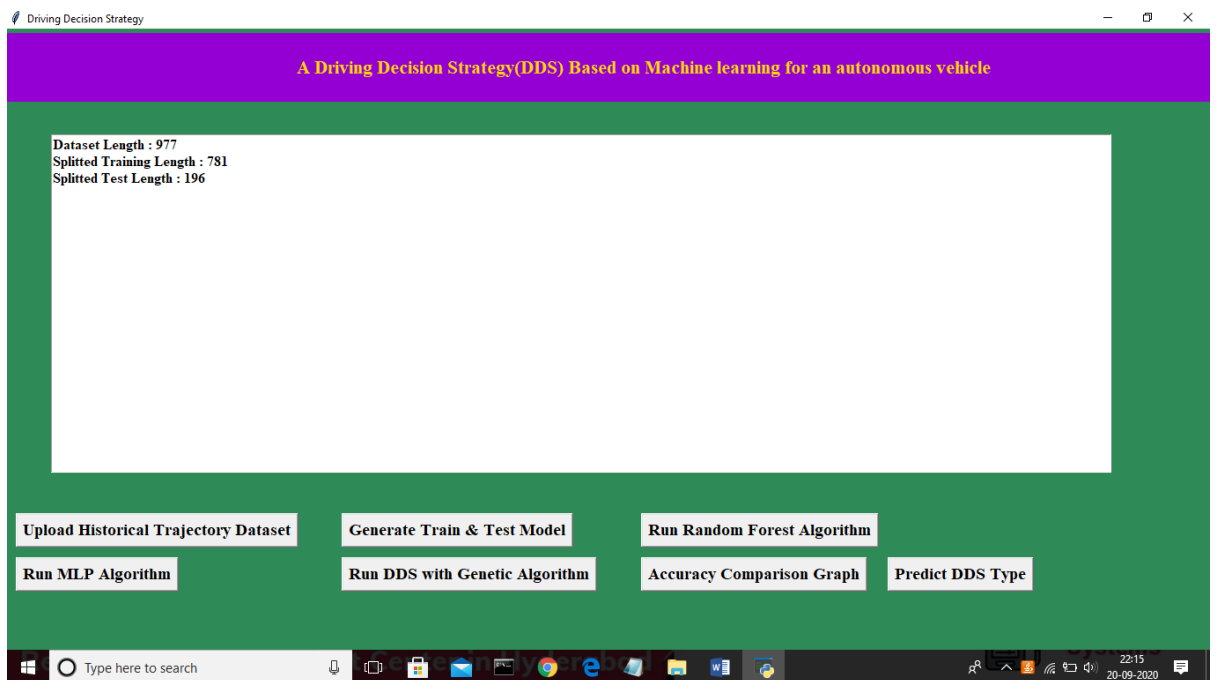
**Fig 5.22 : Output Screen 2**

Now select 'dataset.csv' file and click on 'Open' button to load dataset and to get below screen



**Fig 5.23:** Output Screen 3

In above screen dataset is loaded and now click on ‘Generate Train & Test Model’ button to read dataset and to split dataset into train and test part to generate machine learning train

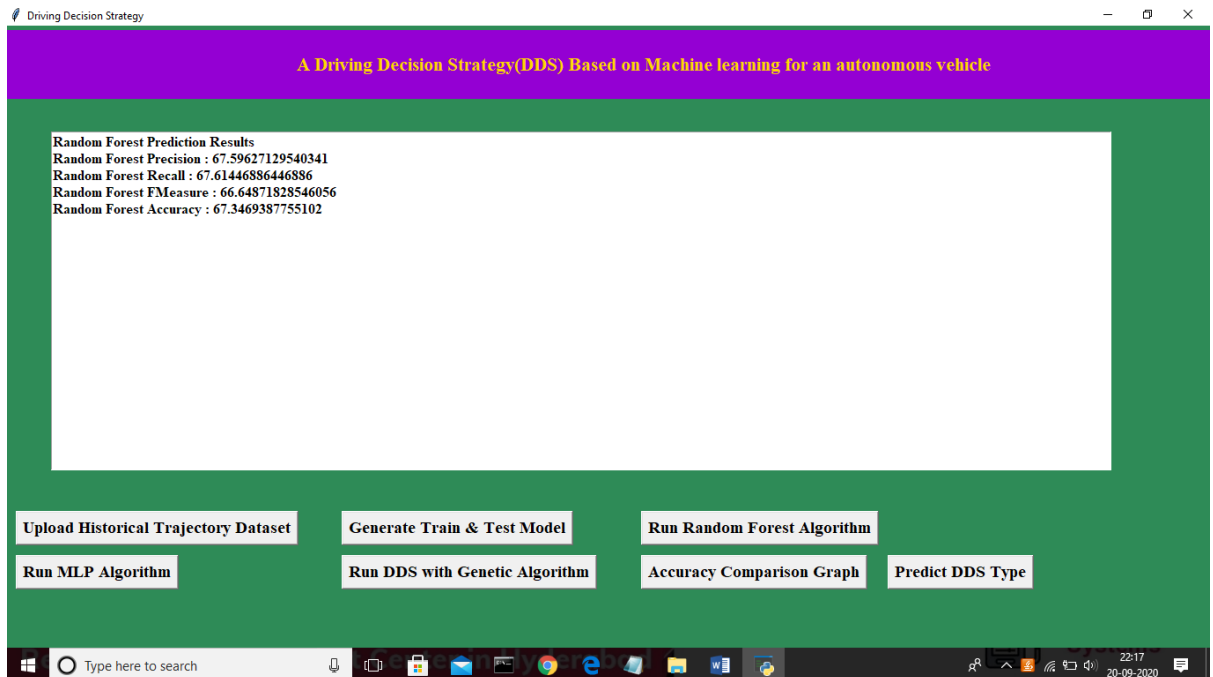


model

**Fig 5.24 :** Output Screen 4

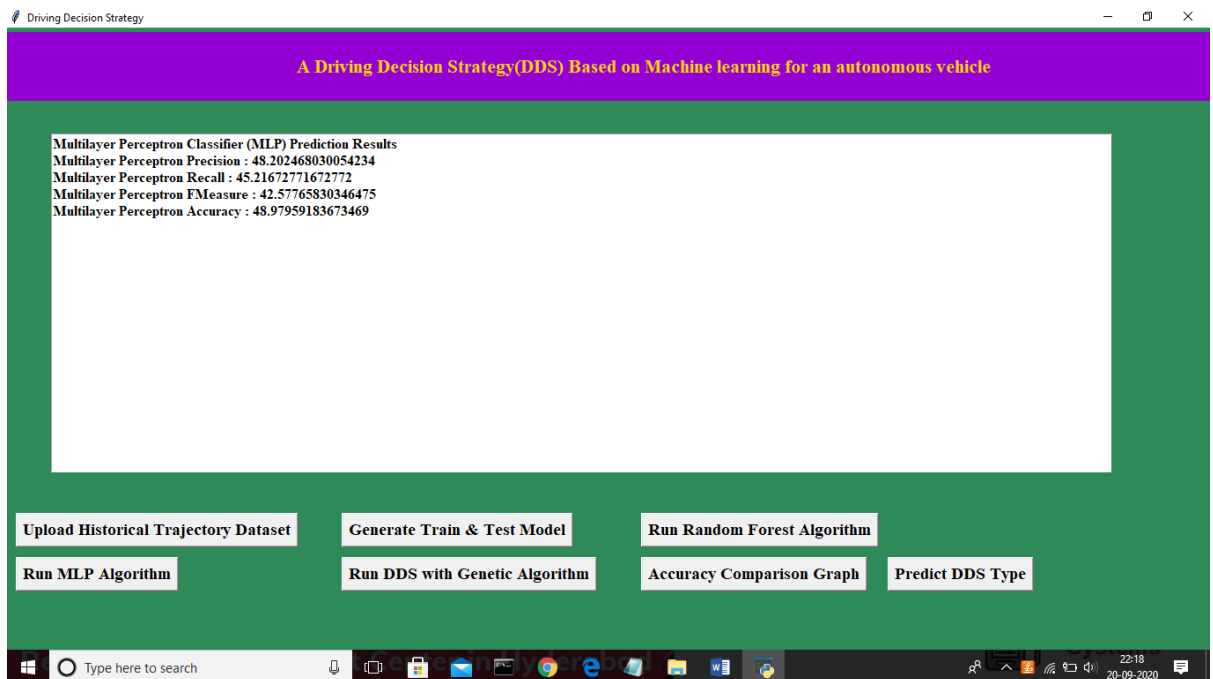
In above screen dataset contains 977 total trajectory records and application using 781 (80% of dataset) records for training and 196 (20% of dataset) for testing.

Now both training and testing data is ready and now click on ‘Run Random Forest Algorithm’ button to train random forest classifier and to calculate its prediction accuracy on 20% test data



**Fig 5.25 : Output Screen 5**

In above screen we calculated random forest accuracy, precision, recall and measure and random forest got 67% prediction accuracy. Now click on ‘Run MLP Algorithm’ button to train MLP model and to calculate its accuracy



**Fig 5.26 : Output Screen 6**

```

DDS.py - E:\(manoj)\September\DrivingDecision\DDS.py (3.7.0)
File Edit Format Run Options Window Help

mfp_precision = precision_score(y_test, prediction_data, average='macro') * 100
mfp_recall = recall_score(y_test, prediction_data, average='macro') * 100
mfp_fmeasure = f1_score(y_test, prediction_data, average='macro') * 100
mfp_acc = accuracy_score(y_test, prediction_data) * 100
text.insert(END, "Multilayer Perceptron Precision : "+str(mfp_precision)+"\n")
text.insert(END, "Multilayer Perceptron Recall : "+str(mfp_recall)+"\n")
text.insert(END, "Multilayer Perceptron FMeasure : "+str(mfp_fmeasure)+"\n")
text.insert(END, "Multilayer Perceptron Accuracy : "+str(mfp_acc)+"\n")

def runDDS():
    global classifier
    global dds_acc
    dds = RandomForestClassifier(n_estimators=45, random_state=42)
    selector = GeneticSelectionCV(dds, #algorithm name
                                cv=5,
                                verbose=1,
                                scoring="accuracy",
                                max_features=5,
                                n_populations=10, #population
                                crossover_proba=0.5, #cross over
                                mutation_proba=0.2,
                                n_generations=50,
                                crossover_independent_proba=0.5,
                                mutation_independent_proba=0.05, #mutation
                                tournament_size=3,
                                n_gen_no_change=5,
                                caching=True,
                                n_jobs=-1)

    selector = selector.fit(X_train, y_train)
    text.insert(END, "DDS Prediction Results\n")
    prediction_data = prediction(X_test, selector)

    dds_precision = precision_score(y_test, prediction_data, average='macro') * 100
    dds_recall = recall_score(y_test, prediction_data, average='macro') * 100
    dds_fmeasure = f1_score(y_test, prediction_data, average='macro') * 100
    dds_acc = accuracy_score(y_test, prediction_data) * 100
    text.insert(END, "DDS Precision : "+str(dds_precision)+"\n")
    text.insert(END, "DDS Recall : "+str(dds_recall)+"\n")
    text.insert(END, "DDS FMeasure : "+str(dds_fmeasure)+"\n")
    text.insert(END, "DDS Accuracy : "+str(dds_acc)+"\n")
    classifier = selector

```

In above screen we can see genetic algorithm code used in DDS algorithm and now click on ‘Run DDS with Genetic Algorithm’ button to train DDS and to calculate its prediction accuracy

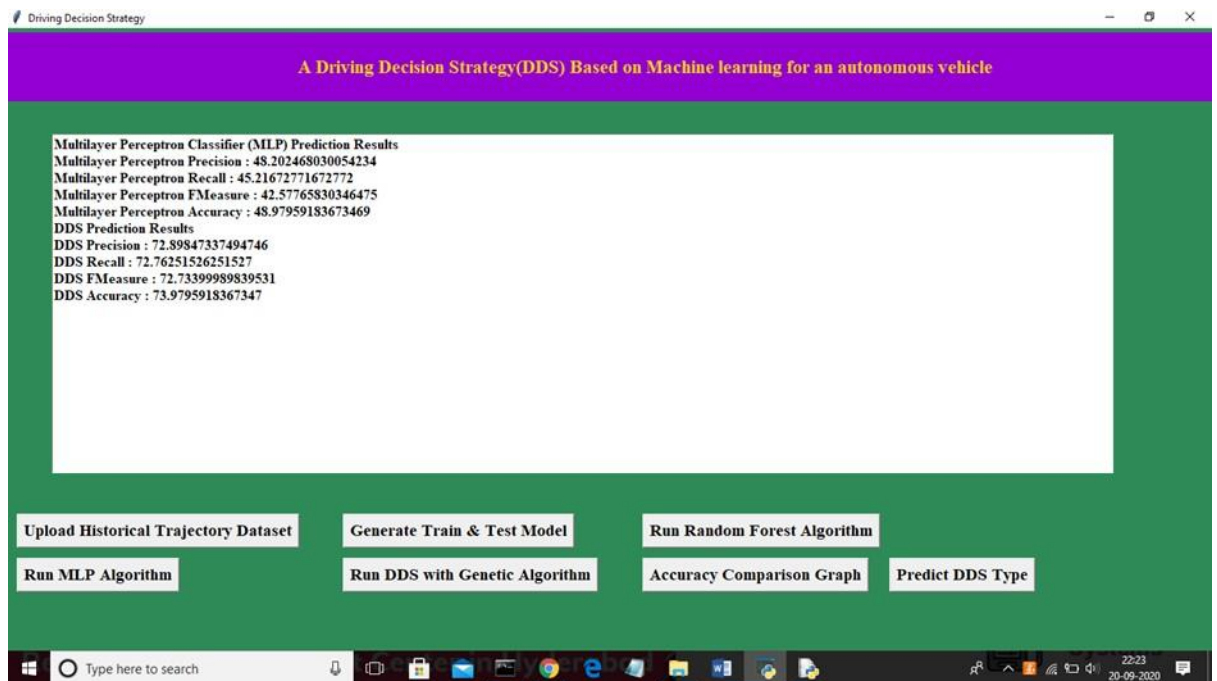
```

C:\Windows\system32\cmd.exe
X: 3.17086421e+00 2.6861317e+00 2.75582645e+01 2.12687006e+00
-2.97549930e-03 -3.28580856e-02 2.42582266e+01 2.67244168e+01, Predicted=1.0
X: 1.55058011e+00 1.43629403e+00 5.69666020e+00 9.28712819e-01
-1.41262342e-02 -3.03407077e-10 2.61683379e+00 4.56782807e-01, Predicted=2.0
X: 5.81618593e+00 4.80726500e+00 2.69147728e+01 4.88038639e+00
1.70513494e-02 5.98727933e-02 5.21970119e+00 1.22686816e+00, Predicted=2.0
X: 3.25334852e+00 3.09307392e+00 4.37115299e+01 2.99738912e+00
-2.10430216e-02 2.79250734e-10 2.12081956e+01 1.62552656e+00, Predicted=1.0
X: 1.43323455e+00 1.32264910e+00 7.6897277e+00 1.00410555e+00
-1.18969963e-02 3.11860405e-03 2.18048280e+00 4.87031814e-01, Predicted=2.0
X: 8.72629262e+00 5.35386515e+00 6.09335735e+01 9.54408440e+00
3.12105289e-02 -7.44929275e-02 1.06421152e+01 2.99358712e+00, Predicted=2.0
X: 8.95650553 6.59108107 41.89406723 9.86112013 -0.33406502 -0.30349898
20.94779482 7.51610055, Predicted=0.0
X: 3.56045576e+00 3.44734689e+00 2.94728044e+01 2.54494455e+00
1.98276771e-02 -3.85906067e-02 2.60159650e+01 2.66632875e+00, Predicted=1.0
X: 2.63091370e+00 2.89080550e+00 2.45629191e+01 1.71683441e+00
-7.5906815e-03 5.25934613e-03 1.05878266e+01 8.27121287e-01, Predicted=1.0
X: 3.34300906e+00 3.25834140e+00 3.94719265e+01 2.92674639e+00
2.75700467e-02 -8.45964339e-08 3.34896920e+01 2.89760266e+00, Predicted=1.0
X: 1.95390303e+00 9.84378016e-01 6.76323176e+01 3.62304746e+00
9.06720300e-03 -6.33315036e-04 1.28830165e+01 6.27824730e-01, Predicted=1.0
Selecting features with genetic algorithm.
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot be imported from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot be imported from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot be imported from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot be imported from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.feature_selection.base module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.feature_selection. Anything that cannot be imported from sklearn.feature_selection is now part of the private API.
warnings.warn(message, FutureWarning)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.feature_selection.base module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.feature_selection. Anything that cannot be imported from sklearn.feature_selection is now part of the private API.
warnings.warn(message, FutureWarning)

```

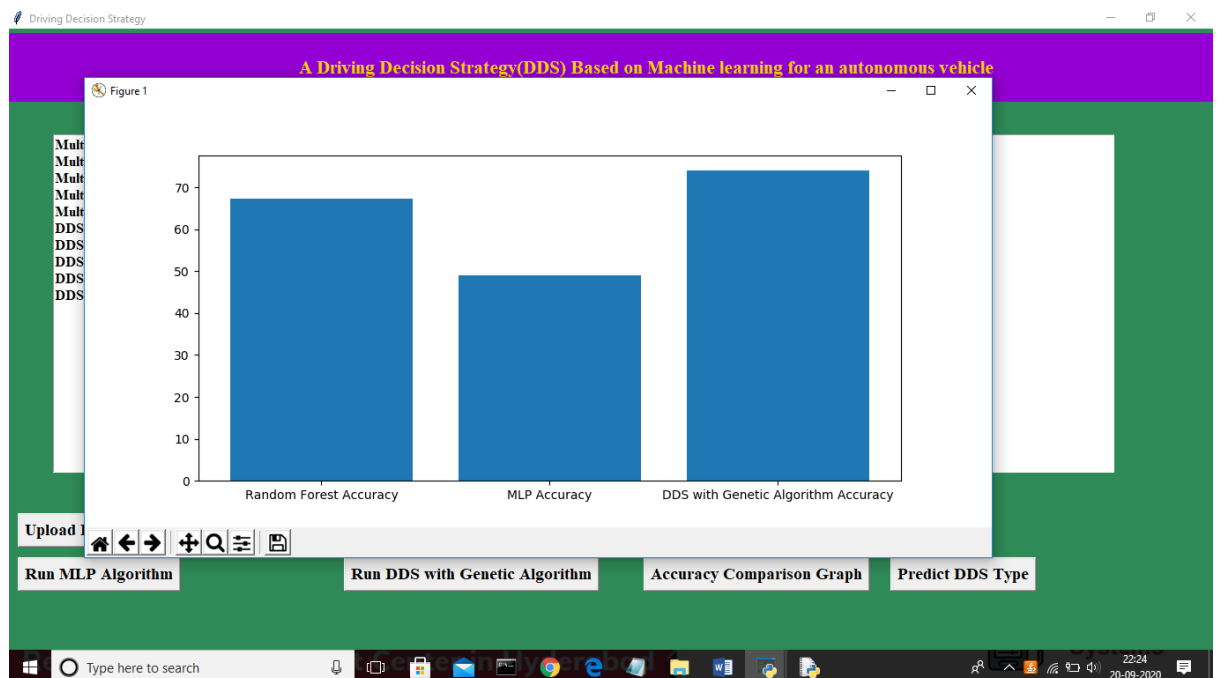
47

In above black console genetic algorithm starts optimal feature selection



**Fig 5.29 : Output Screen 9**

In above screen propose DDS algorithm got 73% prediction accuracy and now click on 'Accuracy Comparison Graph' button to get below graph

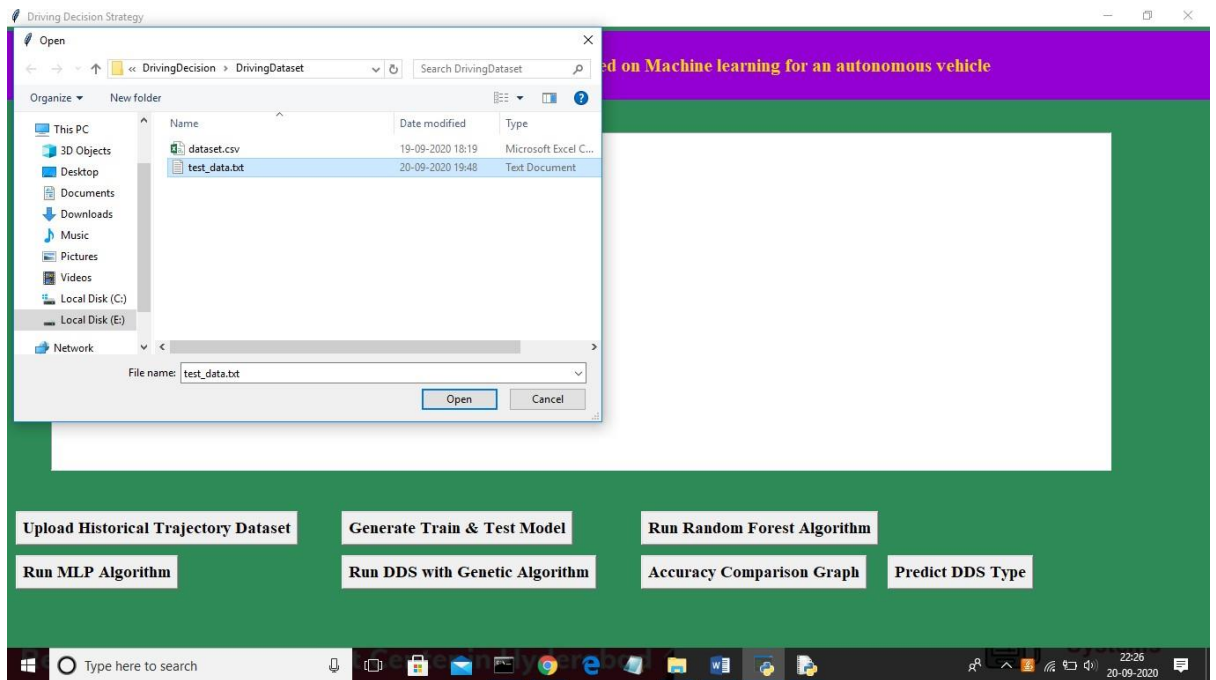


**Fig 5.3.0 : Output Screen 10**

In above graph x-axis represents algorithm name and y-axis represents accuracy of those

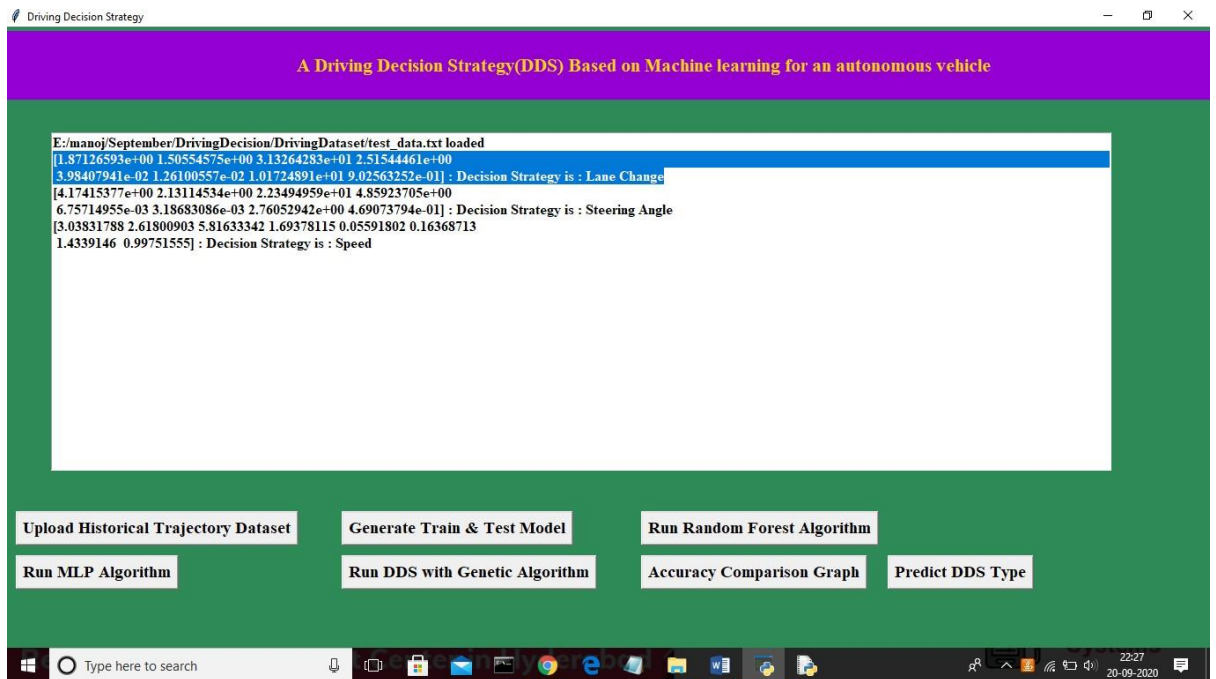


algorithms and from above graph we can conclude that DDS is performing well compare to other two algorithms. Now click on ‘Predict DDS Type’ button to predict test data



**Fig 5.3.1 : Output Screen 11**

In above screen uploading ‘test\_data.txt’ file and click on ‘Open’ button to predict driving decision



**Fig 5.3.2 : Output Screen 12**

In above screen in selected first record we can see decision is Lane Change and for second record values we got decision as 'steering angle' and for third test record we got predicted value as vehicle is in speed mode.

## 5.3 CODES

**DDS.py: -**

```
from tkinter import messagebox from tkinter import *
from tkinter import simpledialog import tkinter
from tkinter import filedialog import matplotlib.pyplot as plt import numpy as np
from tkinter.filedialog import askopenfilename import pandas as pd
from sklearn.model_selection import train_test_split from sklearn.metrics import
accuracy_score
from sklearn.metrics import precision_score from sklearn.metrics import recall_score from
sklearn.metrics import f1_score
from sklearn.neural_network import MLPClassifier from sklearn.ensemble import
RandomForestClassifier from sklearn.preprocessing import LabelEncoder
from genetic_selection import GeneticSelectionCV
main = tkinter.Tk()
main.title("Driving Decision Strategy") #designing main screen
main.geometry("1300x1200")
global filename global X
le = LabelEncoder()
global mlp_acc, rf_acc, dds_acc global classifier
def upload(): #function to driving trajectory dataset
global filename
filename = filedialog.askopenfilename(initialdir="DrivingDataset") text.delete('1.0', END)
text.insert(END,filename+" loaded\n");
def generateTrainTestData():
global X_train, X_test, y_train, y_test text.delete('1.0', END)
train = pd.read_csv(filename) train.drop('trajectory_id', axis=1, inplace=True)
train.drop('start_time', axis=1, inplace=True) train.drop('end_time', axis=1, inplace=True)
print(train)
```



```

train['labels'] = pd.Series(le.fit_transform(train['labels'])) rows = train.shape[0] # gives
number of row count
cols = train.shape[1] # gives number of col count features = cols - 1
print(features)
X = train.values[:, 0:features] Y = train.values[:, features] print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
text.insert(END,"Dataset Length : "+str(len(X))+"\n"); text.insert(END,"Splitted Training
Length : "+str(len(X_train))+"\n"); text.insert(END,"Splitted Test Length :
"+str(len(X_test))+"\n\n");
def prediction(X_test, cls): #prediction done here y_pred = cls.predict(X_test)
for i in range(len(X_test)):
print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
return y_pred
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details): accuracy = accuracy_score(y_test,y_pred)*100
text.insert(END,details+"\n\n")
text.insert(END,"Accuracy : "+str(accuracy)+"\n\n") return accuracy
def runRandomForest(): global rf_acc
global classifier text.delete('1.0', END)
rfc = RandomForestClassifier(n_estimators=2, random_state=0) rfc.fit(X_train, y_train)
text.insert(END,"Random Forest Prediction Results\n") prediction_data = prediction(X_test,
rfc)
random_precision = precision_score(y_test, prediction_data,average='macro') * 100
random_recall = recall_score(y_test, prediction_data,average='macro') * 100
random_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
rf_acc = accuracy_score(y_test,prediction_data)*100 text.insert(END,"Random Forest
Precision : "+str(random_precision)+"\n")
text.insert(END,"Random Forest Recall : "+str(random_recall)+"\n")
text.insert(END,"Random Forest FMeasure : "+str(random_fmeasure)+"\n")
text.insert(END,"Random Forest Accuracy : "+str(rf_acc)+"\n")
classifier = rfc
def runMLP(): global mlp_acc

```

```

text.delete('1.0', END)

cls = MLPClassifier(random_state=1, max_iter=10) cls.fit(X_train, y_train)
text.insert(END, "Multilayer Perceptron Classifier (MLP) Prediction Results\n")
prediction_data = prediction(X_test, cls)

mlp_precision = precision_score(y_test, prediction_data, average='macro') * 100 mlp_recall =
recall_score(y_test, prediction_data, average='macro') * 100 mlp_fmeasure = f1_score(y_test,
prediction_data, average='macro') * 100 mlp_acc =
accuracy_score(y_test, prediction_data) * 100 text.insert(END, "Multilayer Perceptron
Precision : "+str(mlp_precision)+"\n") text.insert(END, "Multilayer Perceptron Recall :
"+str(mlp_recall)+"\n") text.insert(END, "Multilayer Perceptron FMeasure :
"+str(mlp_fmeasure)+"\n") text.insert(END, "Multilayer Perceptron Accuracy :
"+str(mlp_acc)+"\n")

def runDDS(): global classifier global dds_acc
dds = RandomForestClassifier(n_estimators=45, random_state=42) selector =
GeneticSelectionCV(dds, #algorithm name
cv=5, verbose=1,
scoring="accuracy", max_features=5, n_population=10, #population crossover_proba=0.5,
#cross over mutation_proba=0.2, n_generations=50,
crossover_independent_proba=0.5, mutation_independent_proba=0.05, #mutation
tournament_size=3,
n_gen_no_change=5, caching=True,
n_jobs=-1)
selector = selector.fit(X_train, y_train) text.insert(END, "DDS Prediction Results\n")
prediction_data = prediction(X_test, selector)

dds_precision = precision_score(y_test, prediction_data, average='macro') * 100 dds_recall =
recall_score(y_test, prediction_data, average='macro') * 100 dds_fmeasure = f1_score(y_test,
prediction_data, average='macro') * 100 dds_acc =
accuracy_score(y_test, prediction_data) * 100
text.insert(END, "DDS Precision : "+str(dds_precision)+"\n") text.insert(END, "DDS Recall :
"+str(dds_recall)+"\n") text.insert(END, "DDS FMeasure : "+str(dds_fmeasure)+"\n")
text.insert(END, "DDS Accuracy : "+str(dds_acc)+"\n") classifier = selector

def graph():
height = [rf_acc, mlp_acc, dds_acc]

```

```

bars = ('Random Forest Accuracy','MLP Accuracy','DDS with Genetic Algorithm Accuracy')
y_pos = np.arange(len(bars))
plt.bar(y_pos, height) plt.xticks(y_pos, bars) plt.show()
def predictType():
filename = filedialog.askopenfilename(initialdir="DrivingDataset") text.delete('1.0', END)
text.insert(END,filename+" loaded\n"); test = pd.read_csv(filename)
test.drop('trajectory_id', axis=1, inplace=True) test.drop('start_time', axis=1, inplace=True)
test.drop('end_time', axis=1, inplace=True) cols = test.shape[1]
test = test.values[:, 0:cols]
predict = classifier.predict(test) print(predict)
for i in range(len(test)): if predict[i] == 0:
text.insert(END,str(test[i])+" : Decision Strategy is : Lane Change\n")
if predict[i] == 1:
text.insert(END,str(test[i])+" : Decision Strategy is : Speed\n") if predict[i] == 2:
text.insert(END,str(test[i])+" : Decision Strategy is : Steering Angle\n")
font = ('times', 16, 'bold')
title = Label(main, text='A Driving Decision Strategy(DDS) Based on Machine learning for
an autonomous vehicle')
title.config(bg='darkviolet', fg='gold') title.config(font=font) title.config(height=3,
width=120) title.place(x=0,y=5)
font1 = ('times', 12, 'bold') text=Text(main,height=20,width=150) scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set) text.place(x=50,y=120) text.config(font=font1)
font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Historical Trajectory Dataset", command=upload)
uploadButton.place(x=10,y=550)
uploadButton.config(font=font1)
trainButton = Button(main,text="Generate Train &TestModel",
command=generateTrainTestData)
trainButton.place(x=380,y=550) trainButton.config(font=font1)
rfButton = Button(main, text="Run Random Forest Algorithm",
command=runRandomForest)
rfButton.place(x=720,y=550) rfButton.config(font=font1)

```

```

mlpButton = Button(main, text="Run MLP Algorithm", command=runMLP)
mlpButton.place(x=10,y=600)
mlpButton.config(font=font1)
ddsButton = Button(main, text="Run DDS with Genetic Algorithm", command=runDDS)
ddsButton.place(x=380,y=600)
ddsButton.config(font=font1)
graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)
graphButton.place(x=720,y=600)
graphButton.config(font=font1)
predictButton = Button(main, text="Predict DDS Type", command=predictType)
predictButton.place(x=1000,y=600)
predictButton.config(font=font1)
main.config(bg='sea green') main.mainloop()

```

### **test.py:-**

```

import pandas as pd import numpy as np '''
def clean_label(label):
return label.lstrip(',').rstrip(',').replace(',', ' ')
data = pd.read_csv('DrivingDataset/dataset.csv') #data.drop('index', axis=1, inplace=True)
#data.to_csv('mydata.csv', index=False)
df_labeled
data.dropna(subset=['trajectory_id','start_time','end_time','rpm_average','rpm_medium','rpm_
max',' rpm_std','speed_average','speed_medium',
'speed_max','speed_std','labels']) df_labeled.loc[:, 'labels'] = df_labeled['labels'].apply(lambda
x: clean_label(x))
all_labels = df_labeled['labels'].unique() print("Example of trajectory labels:") for label in
all_labels[0:5]:
print(label)
#We can filter out single modal trajectories by taking the labels which do not contain a
comma: single_modality_labels = [elem for elem in all_labels if ',' not in elem]
df_single_modality = df_labeled[df_labeled['labels'].isin(single_modality_labels)]
df_single_modality.to_csv('mydata.csv', index=False)

```

```
mask = np.random.rand(len(df_single_modality)) < 0.7 df_train = df_single_modality[mask]
df_test = df_single_modality[~mask] print(len(df_train))
'''
data = pd.read_csv('DrivingDataset/dataset.csv') labels = data.labels.unique()
print(labels)
```

## 6.TESTING & VALIDATION

### **Test:**

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 satisfactory, 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 the software tester has knowledge of the inner workings, structure and language of the software, or at least its 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. 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.



## 7.CONCLUSION

A Driving Decision Strategy was proposed in this paper. It uses a genetic algorithm based on gathered data to establish the vehicle's ideal driving strategy based on the slope and curve of the road it is travelling on, and it visualizes the autonomous vehicle's driving and consumables circumstances to provide drivers. To demonstrate the validity of the DDS, experiments were conducted to determine the optimal driving strategy by evaluating data from an autonomous vehicle. The DDS finds the best driving strategy 40 percent faster than the MLP, despite having similar accuracy. DDS also has a 22 percent higher accuracy than RF and calculates the best driving strategy 20 percent faster than the RF system. When accuracy and real-time are required, the DDS is the best choice. The DDS sends only the data needed to identify the vehicle's optimal driving strategy to the cloud, and analyses it using a genetic algorithm, it is faster than other methods. These tests were carried out in a virtual environment using PCs, which had inadequate visualization capabilities. A real-world test of DDS should be conducted in the future. Expert designers should also improve the visualization components.

Integration of real world scenarios and collection of data from multiple environments will increase the resilience of the DDS. The virtual environment provides valuable information, but the dynamic and unpredictable nature of the real world presents unique challenges. Expanding the scope of the DDS to include factors other than the slope and curvature of the road will result in a more flexible and adaptive driving strategy. For example, factors such as weather, traffic, and the quality of the road surface could be included in the algorithm to optimize performance in a broader range of conditions. Partnering with car manufacturers and industry players is critical for the success of the DDS in real-world autonomous vehicles. Field testing with different model and brand vehicles will help to validate the strategy's effectiveness across different platforms. This will also help standardize autonomous driving strategies and promote interoperability and security in the ever-changing autonomous vehicle environment. Beyond real world testing, research and development efforts should focus on improving the genetic algorithm in the DDS.

## 8.REFERENCES

- [1] Y.N. Jeong, S. R. Son, E.H. Jeong and B. K. Lee, “An Integrated Self Diagnosis System for an Autonomous Vehicle Based on an IoT Gateway and Deep Learning,” *Applied Sciences*, vol. 8, no. 7, July 2018.
- [2] Yukiko Kenmochi, LilianBuzer, Akihiro Sugimoto, Ikuko Shimizu, “Discrete plane segmentation and estimation from a point cloud using local geometric patterns,” *International Journal of Automation and Computing*, vol. 5, no. 3, pp. 246-256, 2008.
- [3] Ning Ye, Yingya Zhang, Ruchuan Wang, Reza Malekian, “Vehicle trajectory prediction based on Hidden Markov Model,” *The KSII Transactions on Internet and Information Systems*, vol. 10, no. 7, 2017.
- [4] Li-Jie Zhao, Tian-You Chai, De-Cheng Yuan, “Selective ensemble extreme learning machine modeling of effluent quality in wastewater treatment plants,” *International Journal of Automation and Computing*, vol. 9, no.6, 2012.