**A Driving Decision Strategy (DDS) Based On**

**Machine Learning For An Autonomous Vehicle**

A PROJECT REPORT

Submitted in partial fulfillment of the Requirement for the award of degree of

**Bachelor of Technology**

Submitted By

**1. Kunduru Shankar Reddy (17B81A1254) 2. Nagala Prabhu Kumar(17B81A1271)**

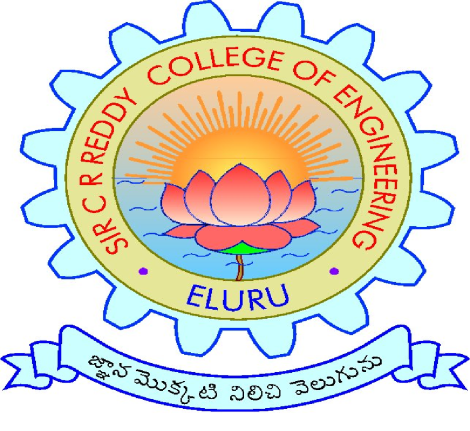
**3. Popuru Sri Harsha(17B81A1286) 4.Marisa Sai Sandhya(17B81A1264)**

*Subm*

Under the esteemed guidance of

**G.VIHARI** Asst.Professor

**Department of Information Technology**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SIR C R REDDY COLLEGE OF ENGINEERING**

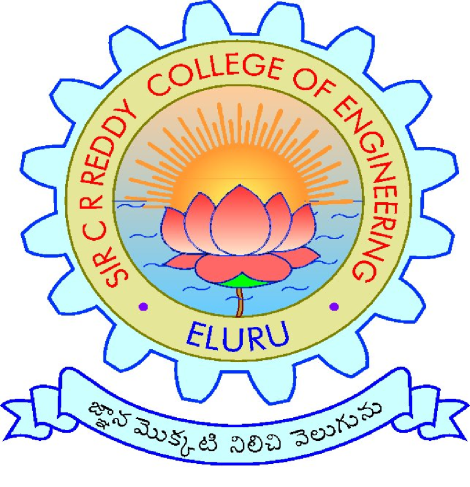
**(Affiliated to Andhra University)**

**ELURU-534007**

**2017-21**

**SIR C R REDDY COLLEGE OF ENGINEERING**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**BONAFIED CERTIFICATE**

This is to certify that the project work titled “**A Driving Decision Strategy(DDS) Based on Machine Learning For An Autonomous Vehicle**” being

Submitted By

**1. Kunduru Shankar Reddy (17B81A1254) 2. Nagala Prabhu Kumar(17B81A1271)**

**3. Popuru Sri Harsha(17B81A1286) 4.Marisa Sai Sandhya(17B81A1264)**

In partial fulfilment for the Bachelor of Technology in Information Technology to the Andhra University is a record of bonafide work carried out under by guidance and supervision. The results embodied in this project report have not been submitted to any other

University or Institute for the award of any degree.

**G.VIHARI,** Asst.Professor **Dr. S. KRISHNA RAO**, PhD

**PROJECT GUIDE HEAD OF THEDEPARTMENT**

DEPARTMENT OF IT DEPARTMENT OF IT

SIR CRR COLLEGE OF ENGG. SIR CRR COLLEGE OF ENGG.

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

First and foremost, I give thanks to Almighty God for all his blessings that he bestowed upon me without which I won’t be where I am today.

We express my sincere thanks to our Principal, Dr. G. Sambasiva Rao for producing me the necessary infrastructure and his guidance in the efficient completion of the project work.

We owe a great deal of gratitude to our beloved Head of the Department Dr. S. Krishna Rao for his continuous support in the journey of my course, for his patience, motivation, enthusiasm and immense knowledge.

We owe a great deal of gratitude to my project guide Mr. G.Vihari for continuous support in the journey of my course, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of work and completing of this project successfully.

Our special thanks to Sri. D.S.R.K.V.Prasad, Librarian and to the entire library staff of Sir C.R.Reddy College of Engineering, for providing the necessary library facilities.

We are also thankful to all the teaching and non-teaching staff of the Department of IT. Last but not least I also express my sincere thanks to all the faculty members of the Department, and my Friends for their help and co-operation in completing the project.

PROJECT MEMBERS

Kunduru Shanker Reddy(17B81A1254)

Nagala Prabhu Kumar(17B81A1271)

Popuru Sri Harsha(17B81A1286)

Marisa Sai Sandhya(17B81A1264)

**A Driving Decision Strategy(DDS)Based on Machine Learning For An Autonomous Vehicle**

**ABSTRACT**

A current autonomous vehicle determines its driving strategy by considering only external factors (Pedestrians, road conditions, etc.) without considering the interior condition of the vehicle. To solve the problem, this paper proposes “A Driving Decision Strategy(DDS) Based on Machine learning for an autonomous vehicle” which determines the optimal strategy of an autonomous vehicle by analyzing not only the external factors, but also the internal factors of the vehicle (consumable conditions, RPM levels etc.). The DDS learns a genetic algorithm using sensor data from vehicles stored in the cloud and determines the optimal driving strategy of an autonomous vehicle. This paper compared the DDS with MLP and RF neural network models to validate the DDS. In the experiment, the DDS had a loss rate approximately 5% lower than existing vehicle gateways and the DDS determined RPM, speed, steering angle and lane changes 40% faster than the MLP and 22% faster than the RF.

# CONTENTS

## Introduction

* 1. Motivation
  2. Existing System
  3. Objective
  4. Outcomes
  5. Applications
  6. Structure of Project
  7. Functional Requirements
  8. Non-Functional Requirements

1. Literature Survey
2. Problem Analysis

3.1 Existing Approach

3.2 Propose System

3.3 Software and Hardware Requirements

3.4 Algorithms

4. System Design

4.1 Use Case Diagram

4.2 Class Diagram

4.3 Sequence Diagram

5. Implementation

5.1 Flow Chart

5.2 Code

6. Testing

7. Results and Discussions

8. Conclusion

9. Future Scope

10. References

**1. INTRODUCTION**

However, as the performance of self-driving cars improves, the number of sensors to recognize data is increasing. An increase in these sensors can cause the in- vehicle overload. Self-driving cars use in-vehicle computers to compute data collected by sensors. As the amount of the computed data increases, it can affect the speed of judgment and control because of overload. These problems can threaten the stability of the vehicle. To prevent the overload, some studies have developed hardware that can perform deep- running operations inside the vehicle, while others use the cloud to compute the vehicle's sensor data. On the other hand, collected from vehicles to determine how the vehicle is driving. This paper proposes a Driving Decision Strategy (DDS) Based on Machine learning for an autonomous vehicle which reduces the in-vehicle computation by generating big data on vehicle driving within the cloud and determines an optimal driving strategy by taking into account the historical data in the cloud. The proposed DDS analyzes them to determine the best driving strategy by using a Genetic algorithm stored in the Cloud

* 1. **Motivation**

Currently, global companies are developing technologies for advanced self-driving cars, which is in the 4th stage. Self driving cars are being developed based on various ICT technologies, and the principle of operation can be classified into three levels of recognition, judgment and control. The recognition step is to recognize and collect information about surrounding situations by utilizing various sensors in vehicles such as GPS, camera, and radar. The judgment step determines the driving strategy based on the recognized information. Then, this step identifies and analyzes the conditions in which the vehicle is placed, and determines the driving plans appropriate to the driving environment and the objectives. The control step determines the speed, direction, etc. about the driving and the vehicle starts driving on its own. An autonomous driving vehicle performs various actions to arrive at its destination, repeating the steps of recognition, judgment and control on its own.

* 1. **Existing System**

k-NN, RF, SVM and Bayes models are existing methods Although studies have been done in the medical field with an advanced data exploration using machine learning algorithms, orthopedic disease prediction is still a relatively new area and must be explored further for the accurate prevention and cure. It mines the double layers of hidden states of vehicle historical trajectories, and then selects the parameters of Hidden Markov Model (HMM) by the historical data. In addition, it uses a Viterbi algorithm to find the double layers hidden states sequences corresponding to the just driven trajectory. Finally, it proposes a new algorithm for vehicle trajectory prediction based on the hidden Markov model of double layers hidden states, and predicts the nearest neighbor unit of location information of the next k stages.

Limitations of existing system: less efficiency and need more are to explored for prevention

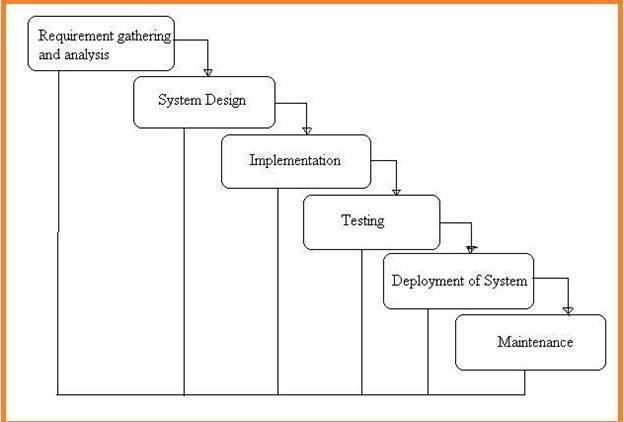
* 1. **Objectives**

The DDS learns a genetic algorithm using sensor data from vehicles stored in the cloud and determines the optimal driving strategy of an autonomous vehicle. This paper compared the DDS with MLP and RF neural network models to validate the DDS. In the experiment, the DDS had a loss rate approximately 5% lower than existing vehicle gateways and the DDS determined RPM, speed, steering angle and lane changes 40% faster than the MLP and 22% faster than the RF.

* 1. **Outcomes**

It 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 Desoto 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

* 1. **STRUCTURE OF PROJECT (SYSTEM ANALYSIS)**



**Fig: 1 Project SDLC**

* + - Project Requisites Accumulating and Analysis
    - Application System Design
    - Practical Implementation
    - Manual Testing of My Application
    - Application Deployment of System
    - Maintenance of the Project

**Requisites Accumulation and Analysis**

It’s the first and foremost stage of the any project as our is a an academic leave for requisites amassing we followed of IEEE Journals and Amassed so many IEEE Relegated papers and final culled a Paper designated “Individual web revisitation by setting and substance importance input and for analysis stage we took referees from the paper and did literature

survey of some papers and amassed all the Requisites of the project in this stage

**System Design**

In System Design has divided into three types like GUI Designing, UML Designing with avails in development of project in facile way with different actor and its utilizer case by utilizer case diagram, flow of the project utilizing sequence, Class diagram gives information about different class in the project with methods that have to be utilized in the project if comes to our project our UML Will utilizable in this way The third and post import for the project in system design is Data base design where we endeavor to design data base predicated on the number of modules in our project

**Implementation**

The Implementation is Phase where we endeavor to give the practical output of the work done in designing stage and most of Coding in Business logic lay coms into action in this stage its main and crucial part of the project

**Testing**

**Unit Testing:** It is done by the developer itself in every stage of the project and fine-tuning the bug and module predicated additionally done by the developer only here we are going to solve all the runtime errors

**Manual Testing:** As our Project is academic Leave, we can do any automatic testing so we follow manual testing by endeavor and error methods

**Deployment of System and Maintenance**

Once the project is total yare, we will come to deployment of client system in genuinely world as its academic leave we did deployment i our college lab only with all need Software’s with having Windows OS.

The Maintenance of our Project is one-time process only

* 1. **Functional Requirements**
     + Data Collection
     + Data Preprocessing
     + Training and Testing
     + Modeling
     + Predicting
  2. **Non Functional Requirements**

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are> 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**Examples of Non-Functional Requirements**

Here, are some examples of non-functional requirement:

* + - Users must upload dataset
    - The software should be portable. So moving from one OS to other OS does not create any problem.
    - Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

**Advantages of Non-Functional Requirement**

Benefits/pros of Non-functional testing are:

* + - The nonfunctional requirements ensure the software system follow legal and compliance rules.
    - They ensure the reliability, availability, and performance of the software system
    - They ensure good user experience and ease of operating the software.
    - They help in formulating security policy of the software system.

**Disadvantages of Non-Functional Requirement**

Cons/drawbacks of Non-function requirement are:

* + - None functional requirement may affect the various high-level software subsystem
    - They require special consideration during the software architecture/high-level design phase which increases costs.
    - Their implementation does not usually map to the specific software sub-system,
    - It is tough to modify non-functional once you pass the architecture phase.

**Key Learning**

The character of the time period, the length of road, the weather, the bus speed and the rate of road usage are adopted as input vectors in Support Vector Machine

# LITERATURE SURVEY

This paper proposes “An Integrated Self-diagnosis System (ISS) for an Autonomous Vehicle that collects information from the sensors of an autonomous vehicle, diagnoses itself, and the influence between its parts by using Deep Learning and informs the driver of the result. The ISS consists of three modules. The first In-Vehicle Gateway Module (In-VGM) collects the data from the in-vehicle sensors, consisting of media data like a black box, driving radar, and the control messages of the vehicle, and transfers each of the data collected through each Controller Area Network (CAN), FlexRay, and Media Oriented Systems Transport (MOST) protocols to the on-board diagnostics (OBD) or the actuators. The data collected from the in- vehicle sensors is transferred to the CAN or FlexRay protocol and the media data collected while driving is transferred to the MOST protocol. Various types of messages transferred are transformed into a destination protocol message type. The second Optimized Deep Learning Module (ODLM) creates the Training Dataset on the basis of the data collected from the in- vehicle sensors and reasons the risk of the vehicle parts and consumables and the risk of the other parts influenced by a defective part. It diagnoses the vehicle’s total condition risk. The third Data Processing Module (DPM) is based on Edge Computing and has an Edge Computing based Self-diagnosis Service (ECSS) to improve the self-diagnosis speed and reduce the system overhead, while a V2X based Accident Notification Service (VANS) informs the adjacent vehicles and infrastructures of the self-diagnosis result analyzed by the OBD. This paper improves upon the simultaneous message transmission efficiency through the In-VGM by 15.25% and diminishes the learning error rate of a Neural Network algorithm through the ODLM by about 5.5%. Therefore, in addition, by transferring the self-diagnosis information and by managing the time to replace the car parts of an autonomous driving vehicle safely, this reduces loss of life and overall cost.: This paper proposes “An Integrated Self-diagnosis System (ISS) for an Autonomous Vehicle based on an Internet of Things (IoT) Gateway and Deep Learning” that collects information from the sensors of an autonomous vehicle, diagnoses itself, and the influence between its parts by using Deep Learning and informs the driver of the result. The ISS consists of three modules. The first In-Vehicle Gateway Module (In-VGM) collects the data from the in-vehicle sensors, consisting of media data like a black box, driving radar, and the control messages of the vehicle, and transfers each of the data collected through each Controller Area Network (CAN), FlexRay, and Media

Oriented Systems Transport (MOST) protocols to the on-board diagnostics (OBD) or the actuators. The data collected from the in-vehicle sensors is transferred to the CAN or FlexRay, protocol and the media data collected while driving is transferred to the MOST protocol. Various types of messages transferred are transformed into a destination protocol message type. The second Optimized Deep Learning Module (ODLM) creates the Training Dataset on the basis of the data collected from the in-vehicle sensors and reasons the risk of the vehicle parts and consumables and the risk of the other parts influenced by a defective part. It diagnoses the vehicle’s total condition risk. The third Data Processing Module (DPM) is based on Edge Computing and has an Edge Computing based Self-diagnosis Service (ECSS) to improve the self-diagnosis speed and reduce the system overhead, while a V2X based Accident Notification Service (VANS) informs the adjacent vehicles and infrastructures of the self-diagnosis result analyzed by the OBD. This paper improves upon the simultaneous message transmission efficiency through the In-VGM by 15.25% and diminishes the learning error rate of a Neural Network algorithm through the ODLM by about 5.5%. Therefore, in addition, by transferring the self-diagnosis information and by managing the time to replace the car parts of an autonomous driving vehicle safely, this reduces loss of life and overall cost.Vehicle trajectory prediction can not only provide accurate location-based services, but also can monitor and predict traffic situation in advance, and then further recommend the optimal route for users. In this paper, firstly, we mine the double layers of hidden states of vehicle historical trajectories, and then determine the parameters of HMM (hidden Markov model) by historical data. Secondly, we adopt Viterbi algorithm to seek the double layers hidden states sequences corresponding to the just driven trajectory. Finally, we propose a new algorithm (DDS) for vehicle trajectory prediction based on the hidden Markov model of double layers hidden states, and predict the nearest neighbor unit of location information of the next k stages. The experimental results demonstrate that the prediction accuracy of the proposed algorithm is increased by 18.3% compared with TPMO algorithm and increased by 23.1% compared with Naive algorithm in aspect of predicting the next k phases' trajectories.

# PROBLEM ANALYSIS

* 1. **Existing Approach:**

k-NN, RF, SVM and Bayes models are existing methods Although studies have been done in the medical field with an advanced data exploration using machine learning algorithms, orthopedic disease prediction is still a relatively new area and must be explored further for the accurate prevention and cure.it mines the double layers of hidden states of vehicle historical trajectories, and then selects the parameters of Hidden Markov Model(HMM) by the historical data. In addition, it uses a Viterbi algorithm to find the double layers hidden states sequences corresponding to the just driven trajectory. Finally, it proposes a new algorithm for vehicle trajectory prediction based on the hidden Markov model of double layers hidden states, and predicts the nearest neighbor unit of location information of the next k stages.

Drawbacks: less efficiency and need more are to explored for prevention

* 1. **Proposed System**

Here we proposes “A Driving Decision Strategy(DDS) Based on Machine learning for an autonomous vehicle” which determines the optimal strategy of an autonomous vehicle by analyzing not only the external factors, but also the internal factors of the vehicle (consumable conditions, RPM levels etc.). The DDS learns a genetic algorithm using sensor data from vehicles stored in the cloud and determines the optimal driving strategy of an autonomous vehicle. This paper compared the DDS with MLP and RF neural network models to validate the DDS. In the experiment, the DDS had a loss rate approximately 5% lower than existing vehicle gateways and the DDS determined RPM, speed, steering angle and lane changes 40% faster than the MLP and 22% faster than the RF.

**Advantages:** These improvements system to control the vehicle based on sensor data

* 1. **Software and Hardware Requirements Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

* + - Python idle 3.7 version
    - Anaconda 3.7
    - Jupiter

**Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system : windows, linux

Processor : minimum intel i3

Ram : minimum 4 gb

Hard disk : minimum 250gb

* 1. **Algorithms**

Random Forest: Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

MLP: A multilayer perceptron (MLP) is a class of [feed forward](https://en.wikipedia.org/wiki/Feedforward_neural_network) [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of [perceptron’s](https://en.wikipedia.org/wiki/Perceptron) (with threshold activation). Multilayer perceptron’s are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear [activation function.](https://en.wikipedia.org/wiki/Activation_function) MLP utilizes a [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) technique called [back propagation](https://en.wikipedia.org/wiki/Backpropagation) for training. Its multiple layers and non-linear activation distinguish MLP from a linear [perceptron.](https://en.wikipedia.org/wiki/Perceptron) It can distinguish data that is not [linearly separable.](https://en.wikipedia.org/wiki/Linear_separability)

# SYSTEM DESIGN

**UML Diagrams**

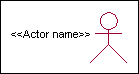
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

**Global Use Case Diagrams:**

Identification of actors:

**Actor:** Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



<<Actor name>>

Actor

An actor is someone or something that: Interacts with or uses the system.

Provides input to and receives information from the system.

Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

Questions to identify actors:

* Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
* Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
* Which external hardware or systems (if any) use the system to perform tasks?
* What problems does this application solve (that is, for whom)?
* And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are: **System Administrator Customer**

**Customer Care**

Identification of use cases:

**Usecase:** A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

**Graphical representation:**



A more detailed description might characterize a use case as:

* Pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the system
* Delivering something of value to the actor Use cases provide a means to
* capture system requirements
* communicate with the end users and domain experts
* test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

* For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
* Name the use cases.
* Describe the use cases briefly by applying terms with which the user is familiar. This makes the description less ambiguous

Questions to identify use cases:

* What are the tasks of each actor?
* Will any actor create, store, change, remove or read information in the system?
* What use case will store, change, remove or read this information?
* Will any actor need to inform the system about sudden external changes?
* Does any actor need to inform about certain occurrences in the system?
* What use cases will support and maintains the system?

**Flow of Events:** A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

* When and how the use case starts and ends
* Use case/actor interactions
* Data needed by the usecase
* Normal sequence of events for the usecase
* Alternate or exceptional flows Construction of Usecase diagrams

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

* actors ("things" outside the system)
* use cases (system boundaries identifying what the system should do)
* Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use cases:

**Communication:** The communication relationship of an actor in a usecase is shown by connecting the actor symbol to the usecase symbol with a solid path. The actor is said to communicate with the usecase.

**Uses:** A Uses relationship between the usecases is shown by generalization arrow from the usecase.

**Extends:** The extend relationship is used when we have one usecase that is similar to another usecase but does a bit more. In essence it is like subclass.

**Sequence Diagrams**

A sequence diagram is a graphical view of a scenario that shows object interaction in a time- based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

**Object:** An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined: An object's concurrency is defined by the concurrency of its class.

**Message:** A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results**.**

**Link:** A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

**Class Diagram:**

Identification of analysis classes: A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items. There are 4 approaches for identifying classes:

1. Noun phrase approach:
2. Common class pattern approach.
3. Use case Driven Sequence or Collaboration approach.
4. Classes , Responsibilities and collaborators Approach

**Noun Phrase Approach:**

The guidelines for identifying the classes:

* Look for nouns and noun phrases in the usecases.
* Some classes are implicit or taken from general knowledge.
* All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
* Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes:
* Adjective classes.

**Common class pattern approach:**

The following are the patterns for finding the candidate classes:

* Concept class.
* Events class.
* Organization class
* Peoples class
* Places class
* Tangible things and devices class.

**Use case driven approach:**

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

**CRC approach:**

The process consists of the following steps:

* Identify classes’ responsibilities (and identify the classes)
* Assign the responsibilities
* Identify the collaborators. Identification of responsibilities of each class:

The questions that should be answered to identify the attributes and methods of a class respectively are:

1. What information about an object should we keep track of?
2. What services must a class provide? Identification of relationships among the classes:

Three types of relationships among the objects are:

* 1. Association: How objects are associated?
  2. Super-sub structure: How are objects organized into super classes and sub classes?
  3. Aggregation: What is the composition of the complex classes?

The questions that will help us to identify the associations are:

* Is the class capable of fulfilling the required task by itself?
* If not, what does it need?
* From what other classes can it acquire what it needs?

Guidelines for identifying the tentative associations:

* A dependency between two or more classes may be an association. Association often

corresponds to a verb or prepositional phrase.

* A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

Some common association patterns are:

* Location association like part of, next to, contained in….. Communication association like talk to, order to ……
* We have to eliminate the unnecessary association like implementation associations, ternary or n- ary associations and derived associations.

Super-sub class relationships: Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class).This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are

**Top-down*:*** Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

**Bottom-up*:*** Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

**Reusability*:*** Move the attributes and methods as high as possible in the hierarchy.

**Multiple inheritances*:*** Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

**Aggregation or a-part-of relationship:**

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn’t behave like its parts. It behaves very difficultly. The major properties of this relationship are transitivity and antisymmetry.

The **questions** whose answers will determine the distinction between the part and whole

relationships are:

* Does the part class belong to the problem domain?
* Is the part class within the system’s responsibilities?
* Does the part class capture more than a single value?( If not then simply include it as an attribute of the whole class)
* Does it provide a useful abstraction in dealing with the problem domain?

There are three types of aggregation relationships. They are:

**Assembly:** It is constructed from its parts and an assembly-part situation physically exists.

**Container:** A physical whole encompasses but is not constructed from physical parts.

**Collection member:** A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

* 1. **Use Case Diagram**

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

Start

Upload Historical Trajectory Dataset



Generate Train & Test Model

Run random forest algorithm

User

Run MLP Algorithm

Run DDS with Genetic Algorithm

Accuracy Graph

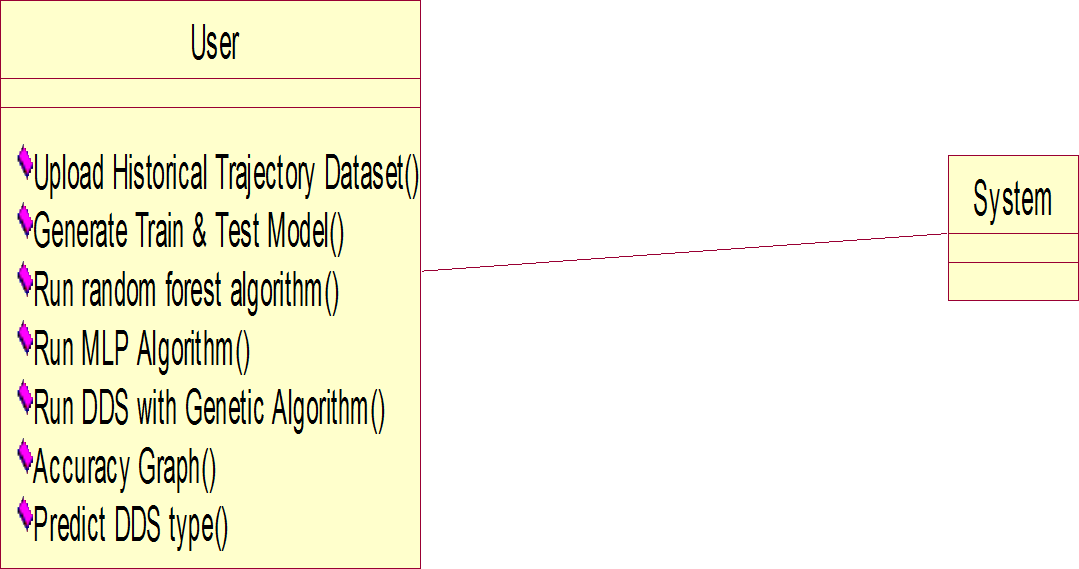
Predict DDS type

Stop

**Fig 1: Use Case Diagram**

* 1. **Class Diagram**

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



**Fig 2: Class Diagram**

* 1. **Sequence Diagram**

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

User

System

Upload Historical Trajectory Dataset

Generate Train & Test Model

Run random forest algorithm

Run MLP Algorithm

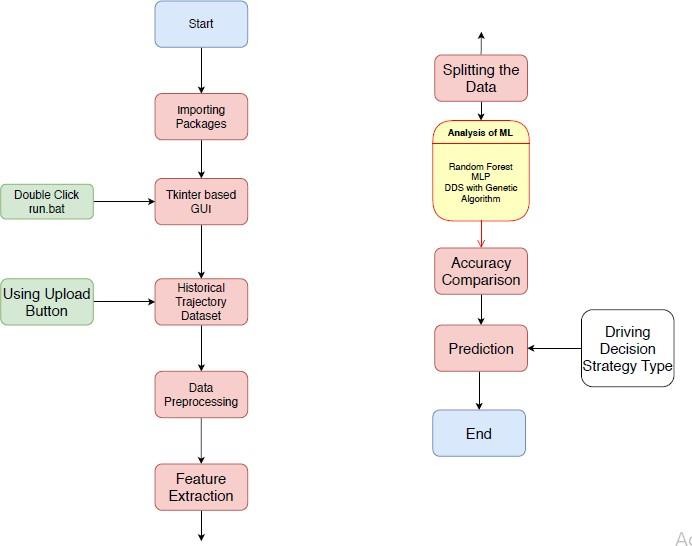
Run DDS with Genetic Algorithm

Accuracy Graph

Predict DDS type

**Fig 3: Sequence Diagram**

1. **IMPLEMENTATION**
   1. **Flow Chart:**



* 1. **Code**

**fromtkinter import messagebox fromtkinter import \* fromtkinter import simpledialog importtkinter**

**fromtkinter import filedialog importmatplotlib.pyplot as plt fromtkinter.filedialog import askopenfilename fromsklearn.model\_selection import train\_test\_split fromsklearn.metrics import accuracy\_score importnumpy as np**

**import pandas as pd**

**fromgenetic\_selection import GeneticSelectionCV fromsklearn.metrics import classification\_report fromsklearn.metrics import confusion\_matrix fromsklearn import svm**

**fromkeras.models import Sequential fromkeras.layers import Dense import time**

**main = tkinter.Tk()**

**main.title("Android Malware Detection") main.geometry("1300x1200")**

**global filename global train**

**globalsvm\_acc, nn\_acc, svmga\_acc, annga\_acc globalX\_train, X\_test, y\_train, y\_test**

**globalsvmga\_classifier globalnnga\_classifier**

**globalsvm\_time,svmga\_time,nn\_time,nnga\_time**

**def upload():**

**global filename**

**filename = filedialog.askopenfilename(initialdir="dataset") pathlabel.config(text=filename)**

**text.delete('1.0', END) text.insert(END,filename+" loaded\n");**

**defgenerateModel():**

**globalX\_train, X\_test, y\_train, y\_test text.delete('1.0', END)**

**train = pd.read\_csv(filename)**

**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 = 0)**

**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)**

**fori in range(len(X\_test)):**

**print("X=%s, Predicted=%s" % (X\_test[i], y\_pred[i])) returny\_pred**

**# Function to calculate accuracy defcal\_accuracy(y\_test, y\_pred, details):**

**cm = confusion\_matrix(y\_test, y\_pred) accuracy = accuracy\_score(y\_test,y\_pred)\*100 text.insert(END,details+"\n\n")**

**text.insert(END,"Accuracy : "+str(accuracy)+"\n\n") text.insert(END,"Report : "+str(classification\_report(y\_test,**

**y\_pred))+"\n")**

**text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n\n") return accuracy**

**defrunSVM():**

**globalsvm\_acc globalsvm\_time start\_time = time.time() text.delete('1.0', END)**

**cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random\_state =**

**2)**

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

**prediction\_data = prediction(X\_test, cls)**

**svm\_acc = cal\_accuracy(y\_test, prediction\_data,'SVM Accuracy')**

**svm\_time = (time.time() - start\_time)**

**defrunSVMGenetic(): text.delete('1.0', END) globalsvmga\_acc globalsvmga\_classifier globalsvmga\_time**

**estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random\_state = 2)**

**svmga\_classifier = GeneticSelectionCV(estimator,**

**cv=5, verbose=1,**

**scoring="accuracy", max\_features=5, n\_population=50, crossover\_proba=0.5, mutation\_proba=0.2, n\_generations=40,**

**crossover\_independent\_proba=0.5, mutation\_independent\_proba=0.05, tournament\_size=3, n\_gen\_no\_change=10, caching=True,**

**n\_jobs=-1) start\_time = time.time()**

**svmga\_classifier = svmga\_classifier.fit(X\_train, y\_train) svmga\_time = svm\_time/2**

**prediction\_data = prediction(X\_test, svmga\_classifier)**

**svmga\_acc = cal\_accuracy(y\_test, prediction\_data,'SVM with GA Algorithm Accuracy, Classification Report & Confusion Matrix')**

**defrunNN():**

**globalnn\_acc globalnn\_time text.delete('1.0', END) start\_time = time.time() model = Sequential()**

**model.add(Dense(4, input\_dim=215, activation='relu')) model.add(Dense(215, activation='relu')) model.add(Dense(1, activation='sigmoid'))**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**model.fit(X\_train, y\_train, epochs=50, batch\_size=64)**

**\_, ann\_acc = model.evaluate(X\_test, y\_test) nn\_acc = ann\_acc\*100**

**text.insert(END,"ANN Accuracy : "+str(nn\_acc)+"\n\n") nn\_time = (time.time() - start\_time)**

**defrunNNGenetic(): globalannga\_acc globalnnga\_time text.delete('1.0', END)**

**train = pd.read\_csv(filename)**

**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[:, 0:100]**

**Y = train.values[:, features] print(Y)**

**X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X, Y, test\_size**

**= 0.2, random\_state = 0) model = Sequential()**

**model.add(Dense(4, input\_dim=100, activation='relu')) model.add(Dense(100, activation='relu')) model.add(Dense(1, activation='sigmoid'))**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**start\_time = time.time() model.fit(X\_train1, y\_train1) nnga\_time = (time.time() - start\_time)**

**\_, ann\_acc = model.evaluate(X\_test1, y\_test1) annga\_acc = ann\_acc\*100**

**text.insert(END,"ANN with Genetic Algorithm Accuracy : "+str(annga\_acc)+"\n\n")**

**def graph():**

**height = [svm\_acc, nn\_acc, svmga\_acc, annga\_acc]**

**bars = ('SVM Accuracy','NNAccuracy','SVM Genetic Acc','NN Genetic Acc')**

**y\_pos = np.arange(len(bars)) plt.bar(y\_pos, height) plt.xticks(y\_pos, bars) plt.show()**

**deftimeGraph():**

**height = [svm\_time,svmga\_time,nn\_time,nnga\_time]**

**bars = ('SVM Time','SVM Genetic Time','NNTime','NN Genetic Time')**

**y\_pos = np.arange(len(bars)) plt.bar(y\_pos, height) plt.xticks(y\_pos, bars) plt.show()**

**font = ('times', 16, 'bold')**

**title = Label(main, text='Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning') #title.config(bg='brown', fg='white')**

**title.config(font=font) title.config(height=3, width=120) title.place(x=0,y=5)**

**font1 = ('times', 14, 'bold')**

**uploadButton = Button(main, text="Upload Android Malware Dataset", command=upload)**

**uploadButton.place(x=50,y=100) uploadButton.config(font=font1)**

**pathlabel = Label(main) pathlabel.config(bg='brown', fg='white') pathlabel.config(font=font1) pathlabel.place(x=460,y=100)**

**generateButton = Button(main, text="Generate Train & Test Model", command=generateModel)**

**generateButton.place(x=50,y=150) generateButton.config(font=font1)**

**svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)**

**svmButton.place(x=330,y=150) svmButton.config(font=font1)**

**svmgaButton = Button(main, text="Run SVM with Genetic Algorithm", command=runSVMGenetic)**

**svmgaButton.place(x=540,y=150) svmgaButton.config(font=font1)**

**nnButton = Button(main, text="Run Neural Network Algorithm", command=runNN)**

**nnButton.place(x=870,y=150) nnButton.config(font=font1)**

**nngaButton = Button(main, text="Run Neural Network with Genetic Algorithm", command=runNNGenetic) nngaButton.place(x=50,y=200)**

**nngaButton.config(font=font1)**

**graphButton = Button(main, text="Accuracy Graph", command=graph) graphButton.place(x=460,y=200)**

**graphButton.config(font=font1)**

**exitButton = Button(main, text="Execution Time Graph", command=timeGraph)**

**exitButton.place(x=650,y=200) exitButton.config(font=font1)**

**font1 = ('times', 12, 'bold') text=Text(main,height=20,width=150) scroll=Scrollbar(text) text.configure(yscrollcommand=scroll.set) text.place(x=10,y=250) text.config(font=font1)**

**#main.config() main.mainloop()**

**6.1 Software Testing**

1. **TESTING**

**Testing:** Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

**Types of Testing**

* 1. White Box Testing
  2. Black Box Testing
  3. Unit testing
  4. Integration Testing
  5. Alpha Testing
  6. Beta Testing
  7. Performance Testing and so on

**White Box Testing:** Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers

**Black Box Testing:** A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

**Unit Testing:** Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

**Integration Testing:** The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

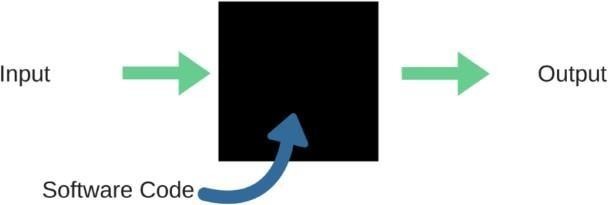
**Alpha Testing:** Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

**Beta Testing:** Final testing before releasing application for commercial purpose. It is

typically done by end- users or others.

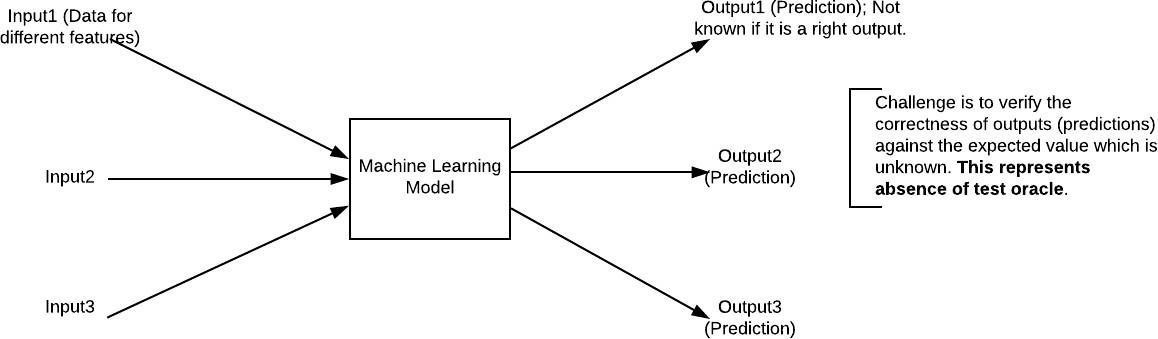
**Performance Testing:** Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

**Black Box Testing:** Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. The below figure represents the black box testing:



**Fig:** Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.



**Fig:** Black Box Testing for Machine Learning algorithms

The above figure represents the black box testing procedure for machine learning algorithms.

**Table.4.1:** Black box Testing

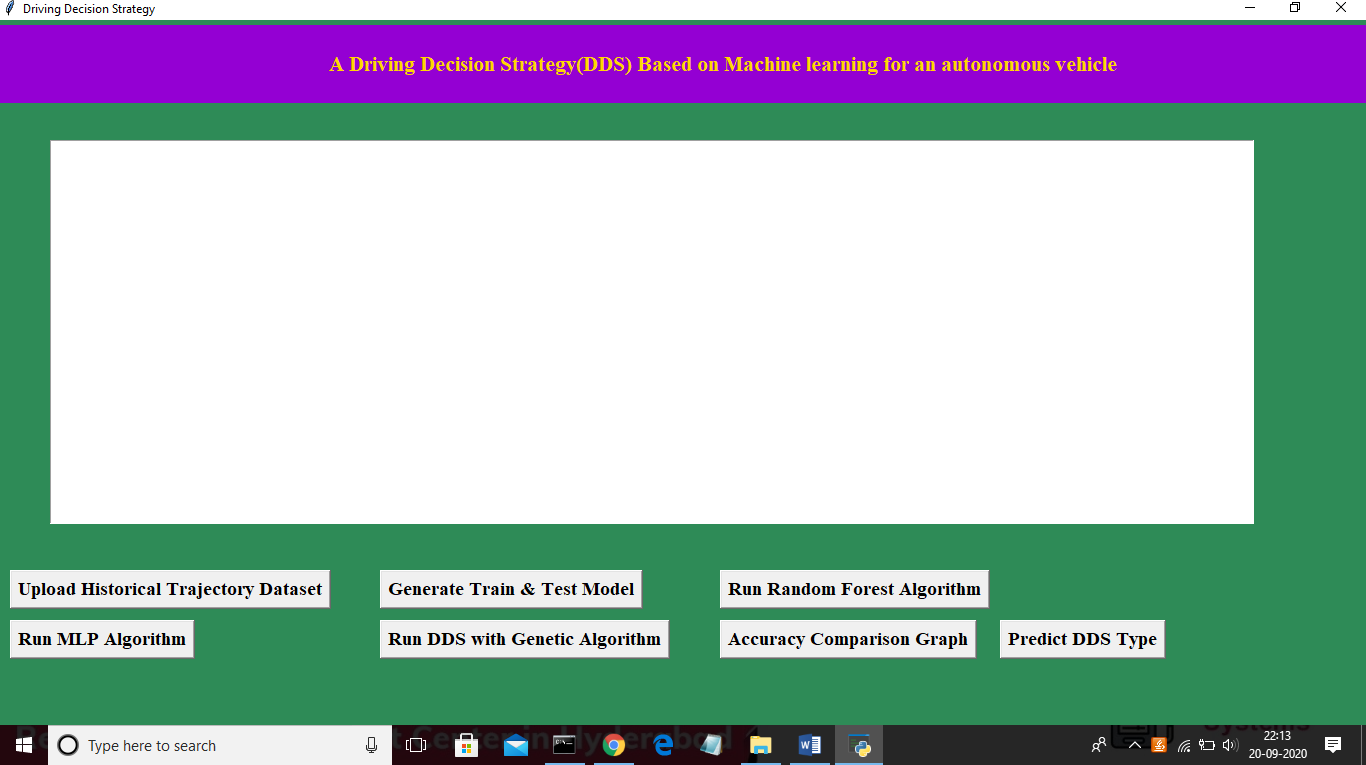
|  |  |  |
| --- | --- | --- |
| **Input** | **Actual Output** | **Predicted Output** |
| [16,6,324,0,0,0,22,0,0,0,0,0,0] | 0 | 0 |
| [16,7,263,7,0,2,700,9,10,1153,832,  9,2] | 1 | 1 |

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or correct program

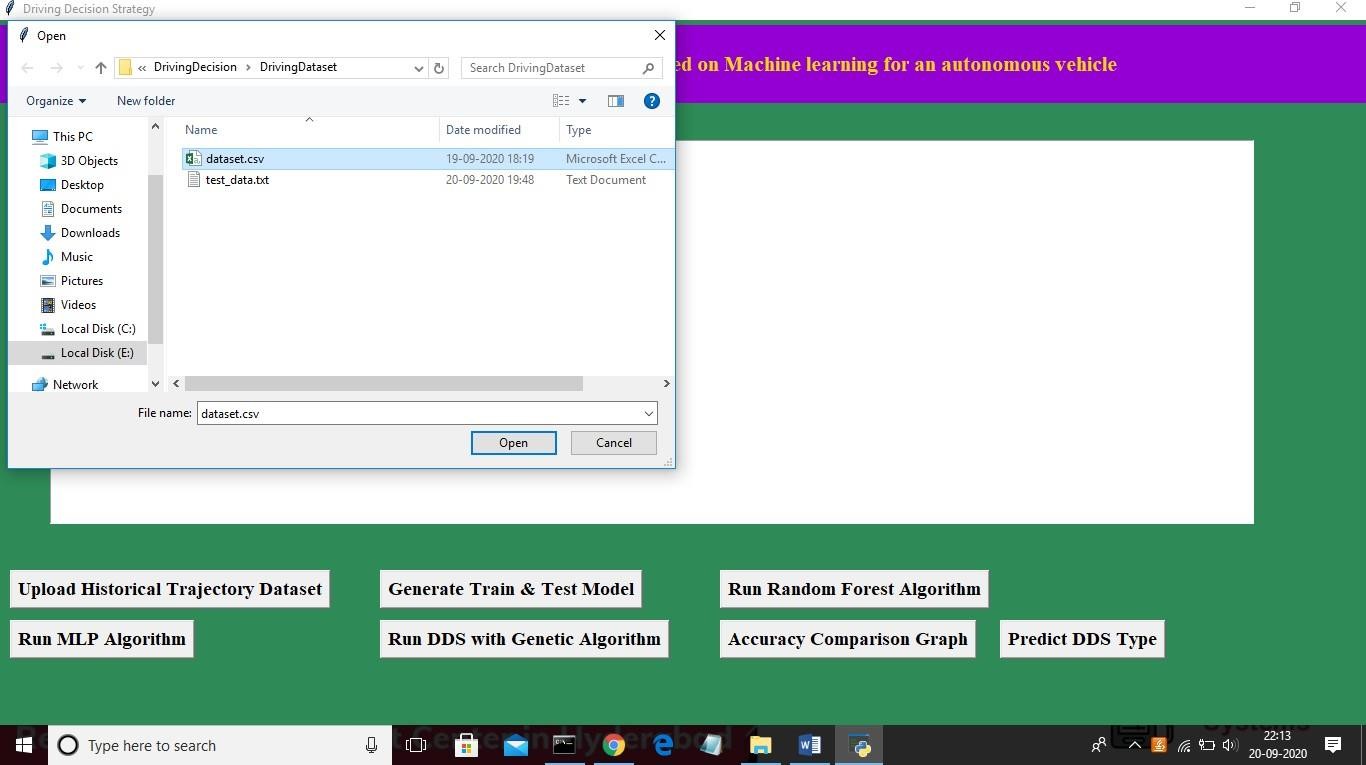
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test | Test Case | Test Case | Test Steps | | | Test | Test |
| Cas | Name | Description | Step | Expected | Actual | Case | Priorit |
| e Id |  |  | Statu | Y |
|  |  |  | s |  |
| 01 | Start the | Host the | If it | We | The | High | High |
| Applicatio | application | doesn't | cannot | application |
| N | and test if it | Start | run the | hosts |
|  | starts |  | applicati | success. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | making sure |  | on. |  |  |  |
| the required |
| software is |
| available |
| 02 | Home Page | Check the | If it | We | The | High | High |
| deployment | doesn’t | cannot | application |
| environmen | load. | access | is running |
| t for |  | The | successfully |
| properly |  | applicati | . |
| loading the |  | on. |  |
| application. |  |  |  |
| 03 | User | Verify the | If it | We | The | High | High |
| Mode | working of | doesn’t | cannot | application |
|  | the | Respond | use the | displays the |
|  | application |  | Freestyle | Freestyle |
|  | in freestyle |  | mode. | Page |
|  | mode |  |  |  |
| 04 | Data Input | Verify if the | If it fails | We | The | High | High |
| application | to take the | cannot | application |
| takes input | input or | proceed | updates the |
| and updates | store in | further | input to application |
|  | The |  |
|  | Database |  |  |

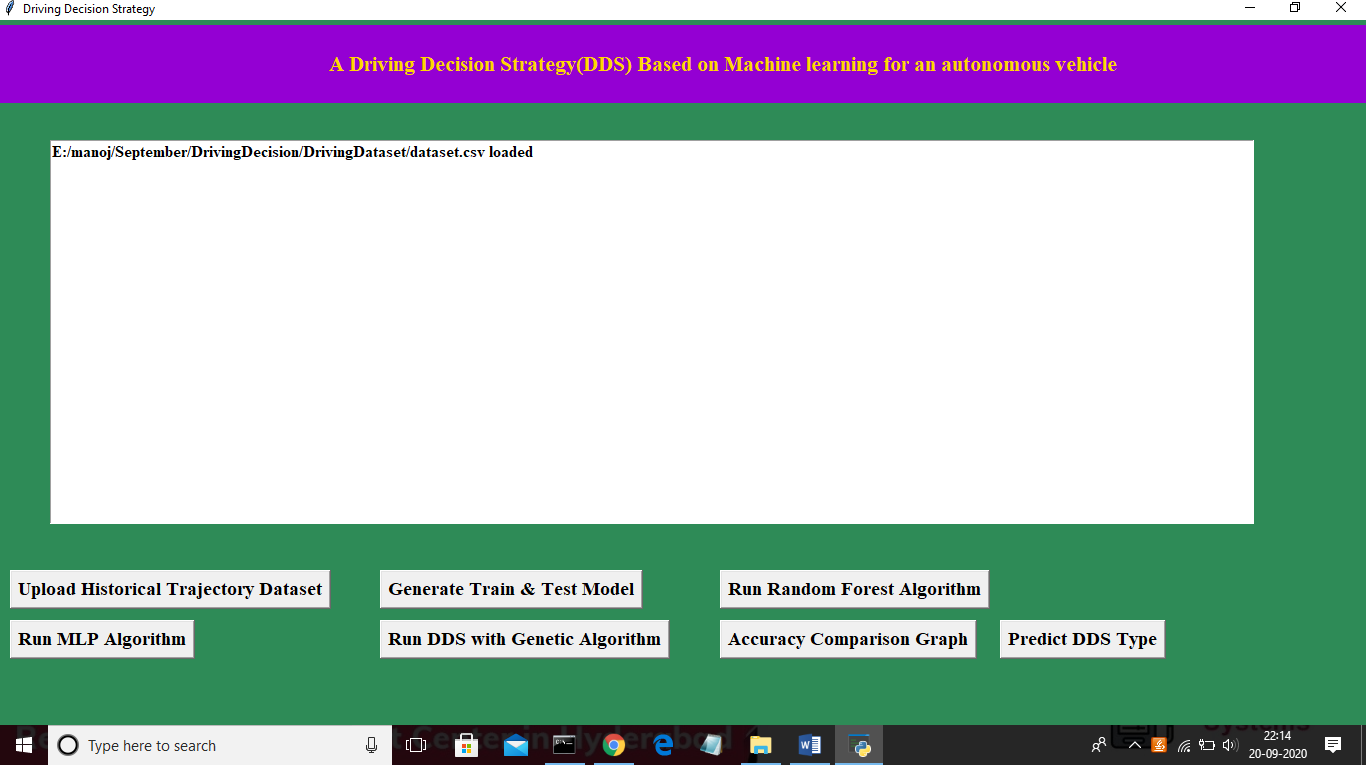
# RESULTS AND DISCUSSIONS



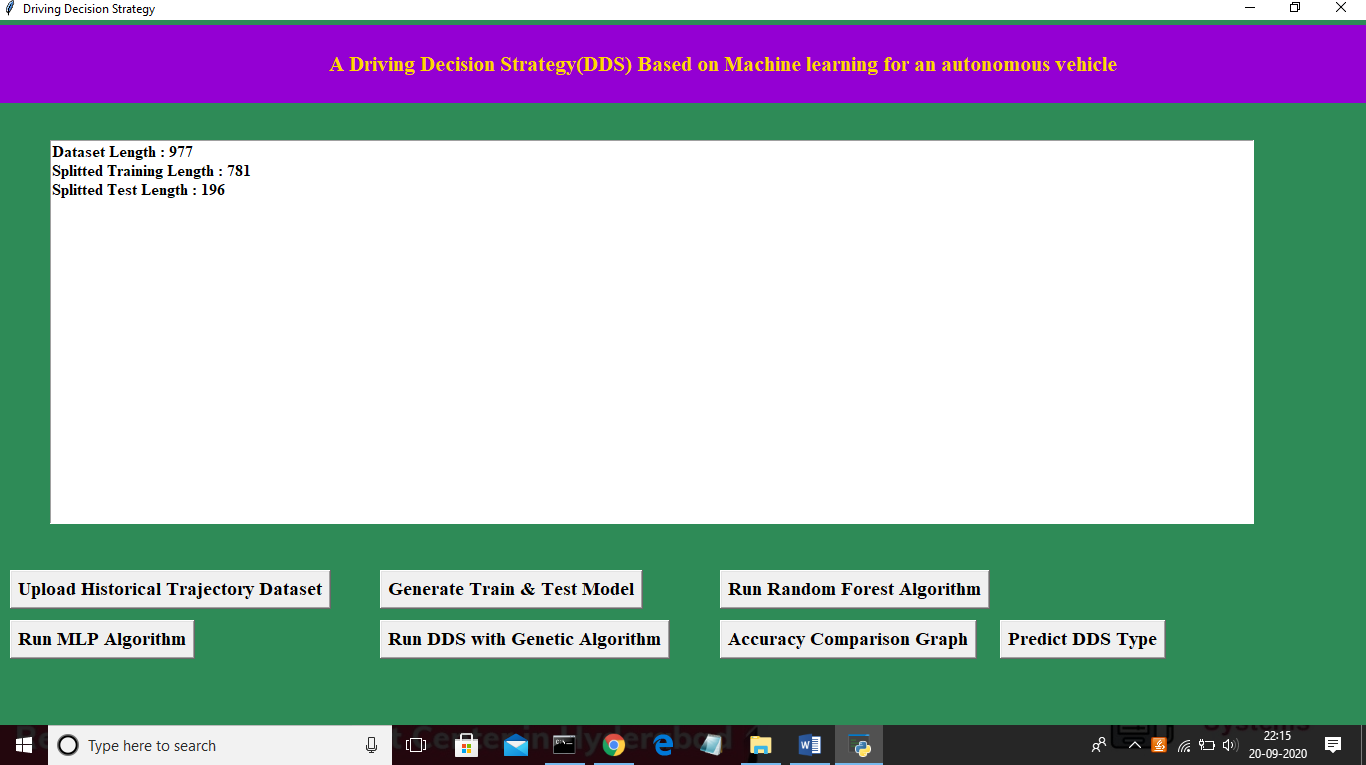
In above screen click on ‘Upload Historical Trajectory Dataset’ button and upload dataset



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

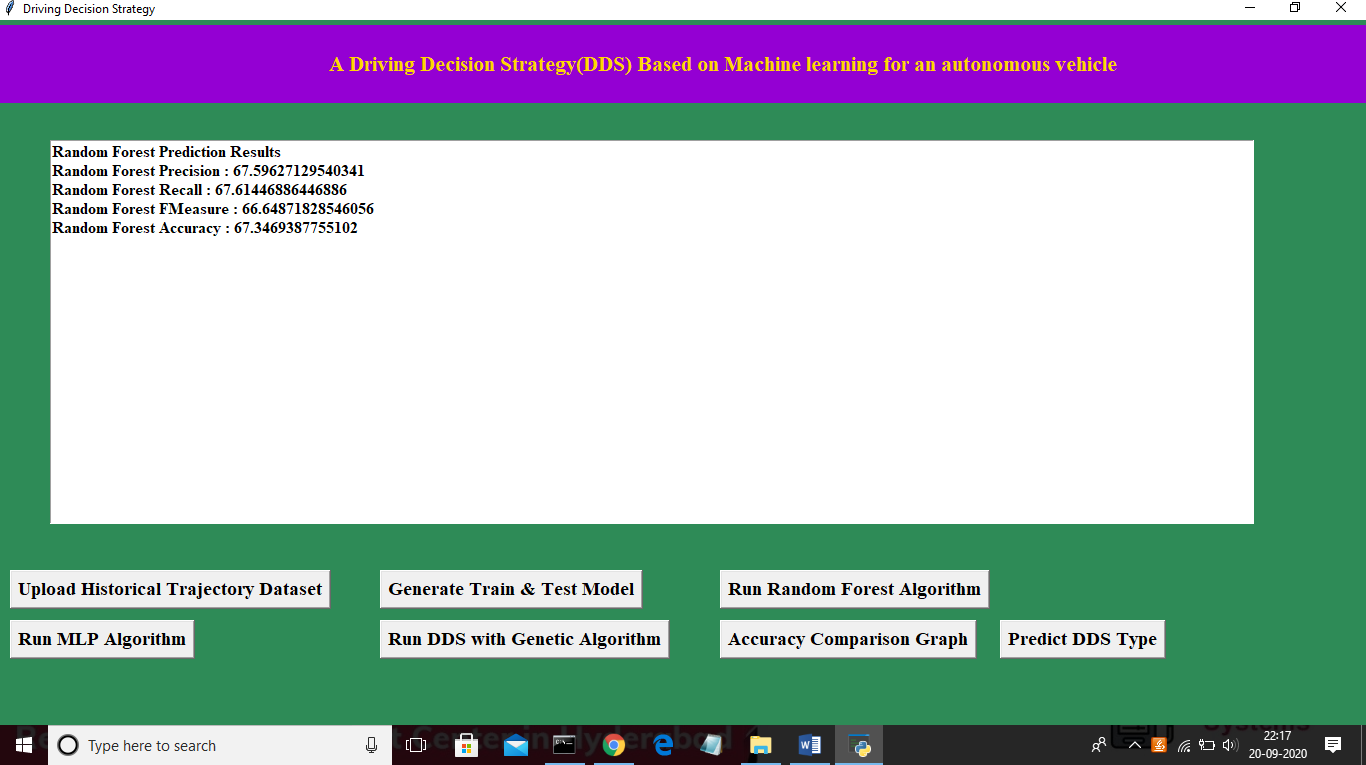


In above screen dataset is loaded and now click on ‘Generate Train & Test Model’ button to read dataset and to split dataset into train and test part to generate machine learning train model

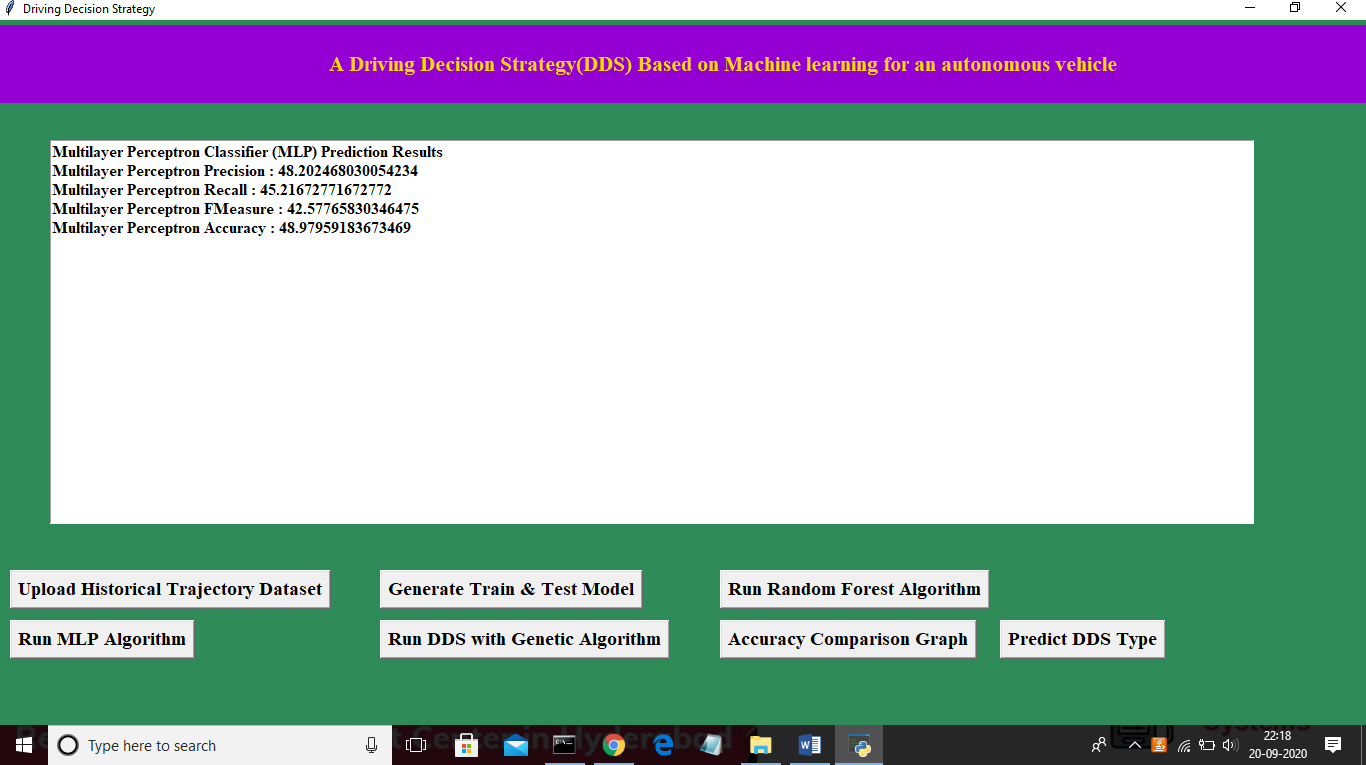


In above screen dataset contains 977 total trajectory records and application using 781 (80% of dataset) records for training and 196 (20% of dataset) for testing. Now both training and testing data is ready and now click on ‘Run Random Forest Algorithm’ button to train

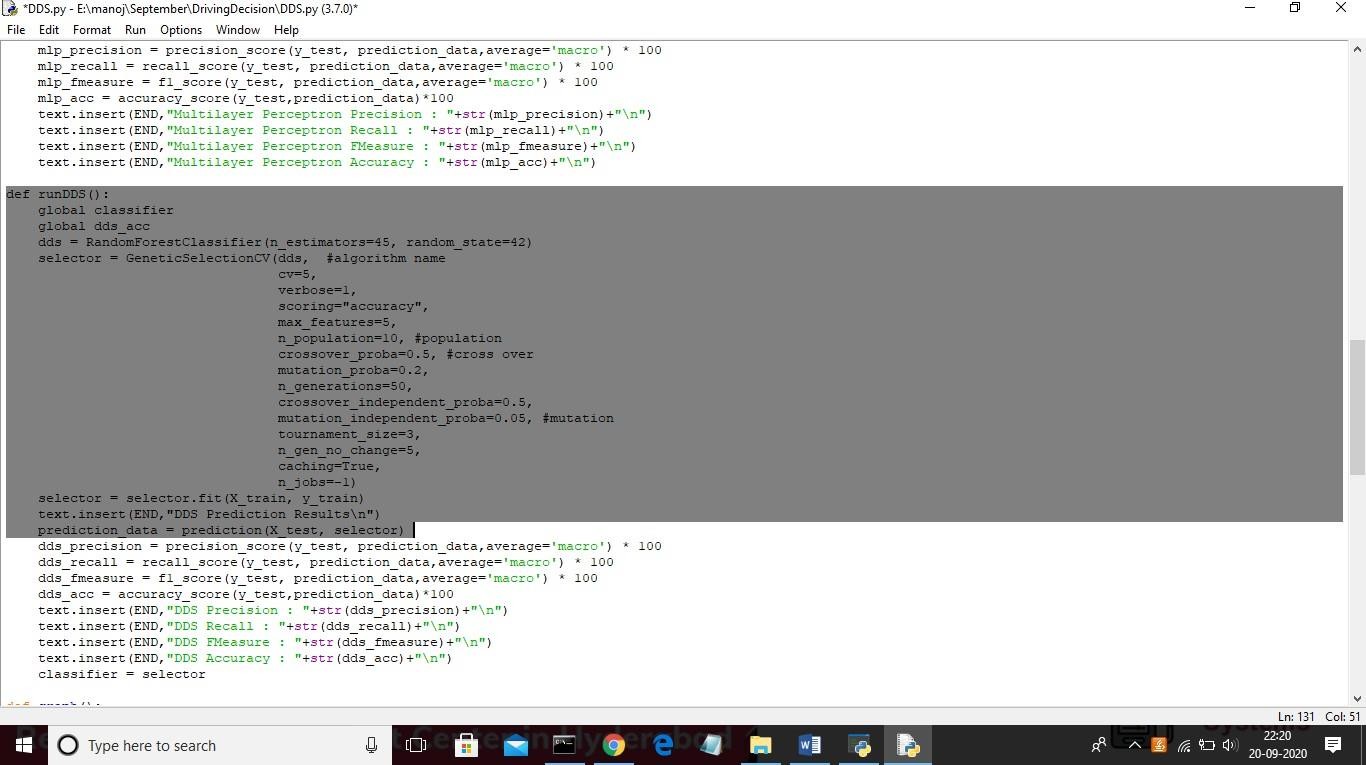
random forest classifier and to calculate its prediction accuracy on 20% test data



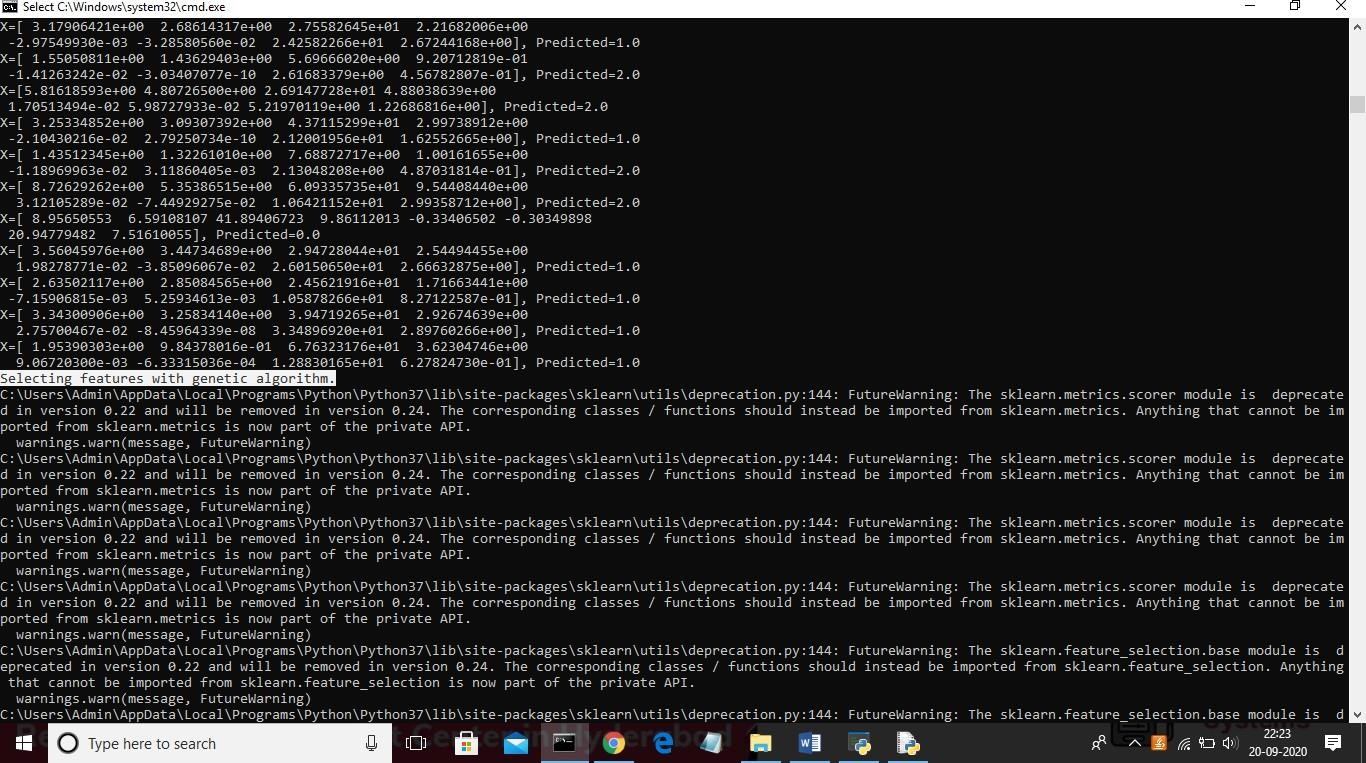
In above screen we calculated random forest accuracy, precision, recall and fmeasure and random forest got 67% prediction accuracy. Now click on ‘Run MLP Algorithm’ button to train MLP model and to calculate its accuracy



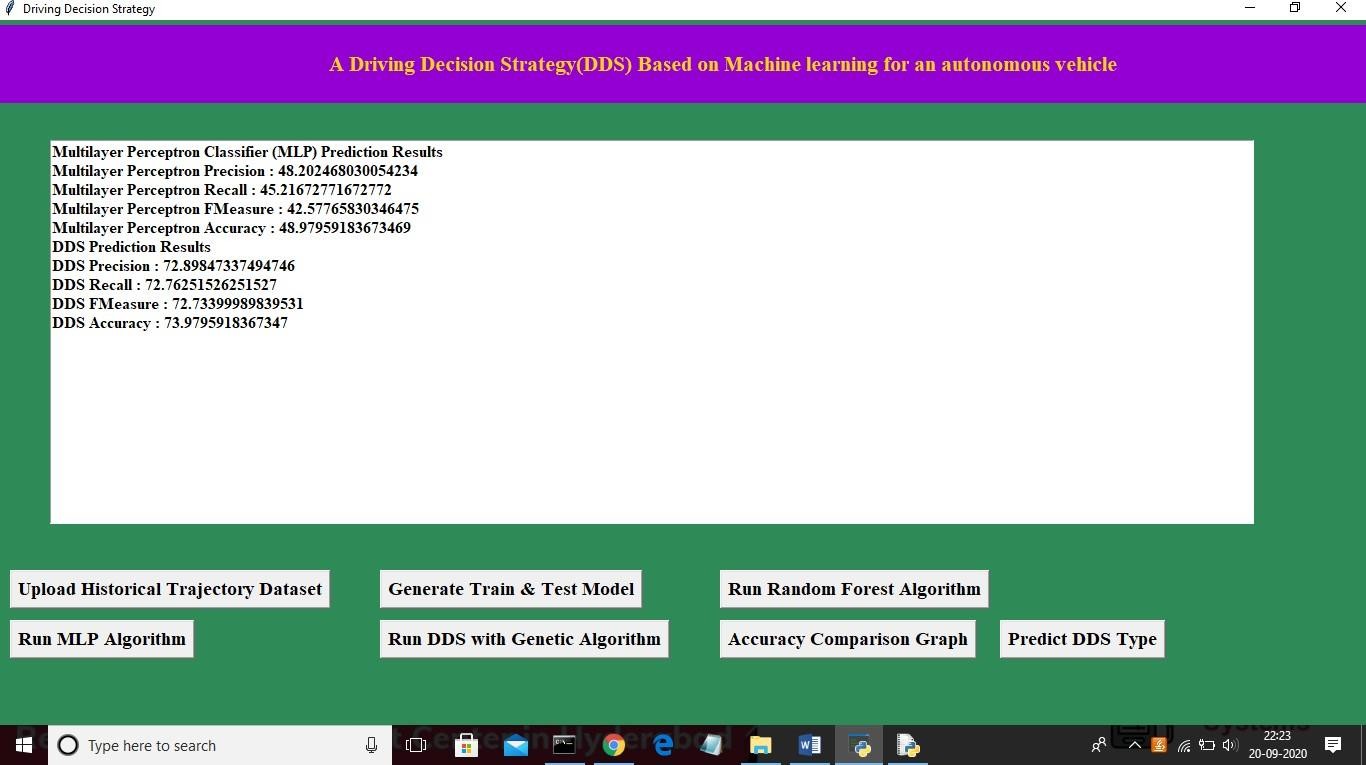
In above screen MLP got 48% prediction accuracy and in below screen we can see genetic algorithm code used for building propose DDS algorithm



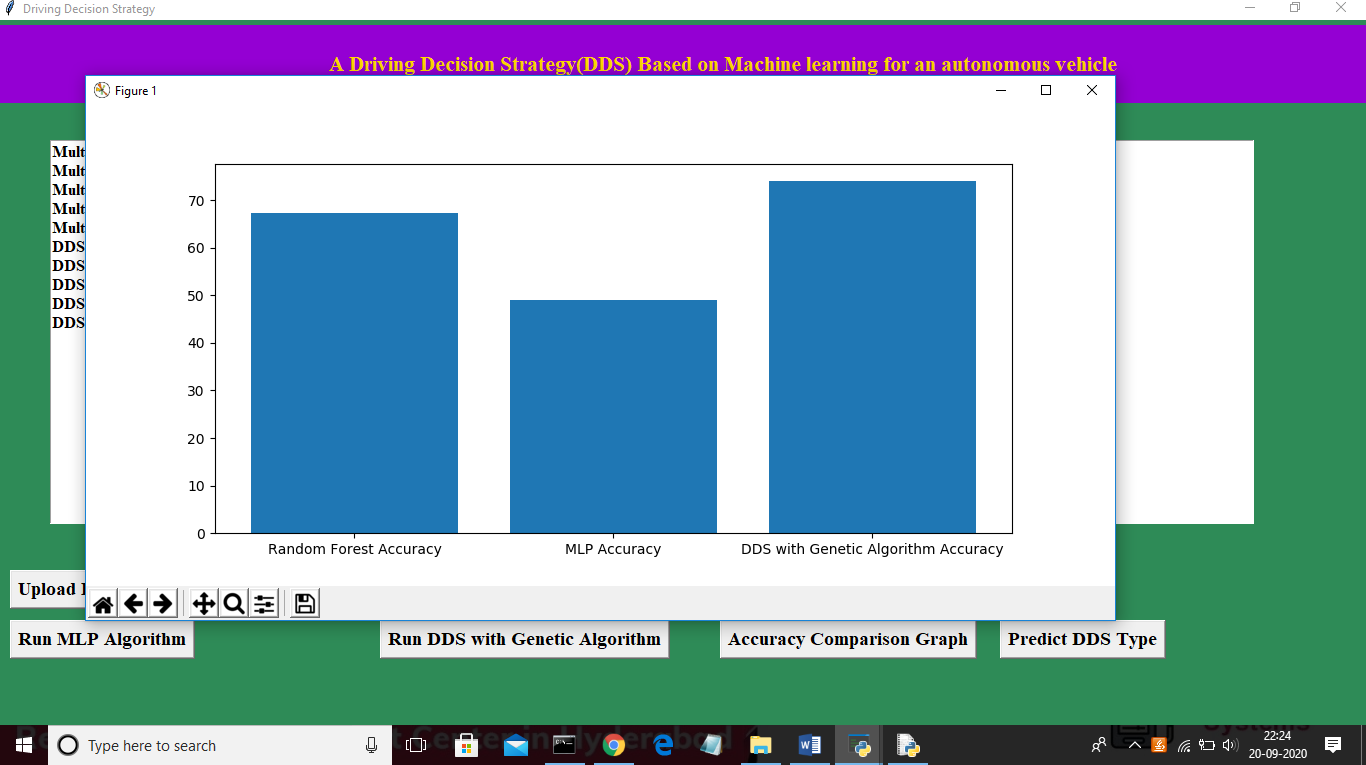
In above screen we can see genetic algorithm code used in DDS algorithm and now click on ‘Run DDS with Genetic Algorithm’ button to train DDS and to calculate its prediction accuracy



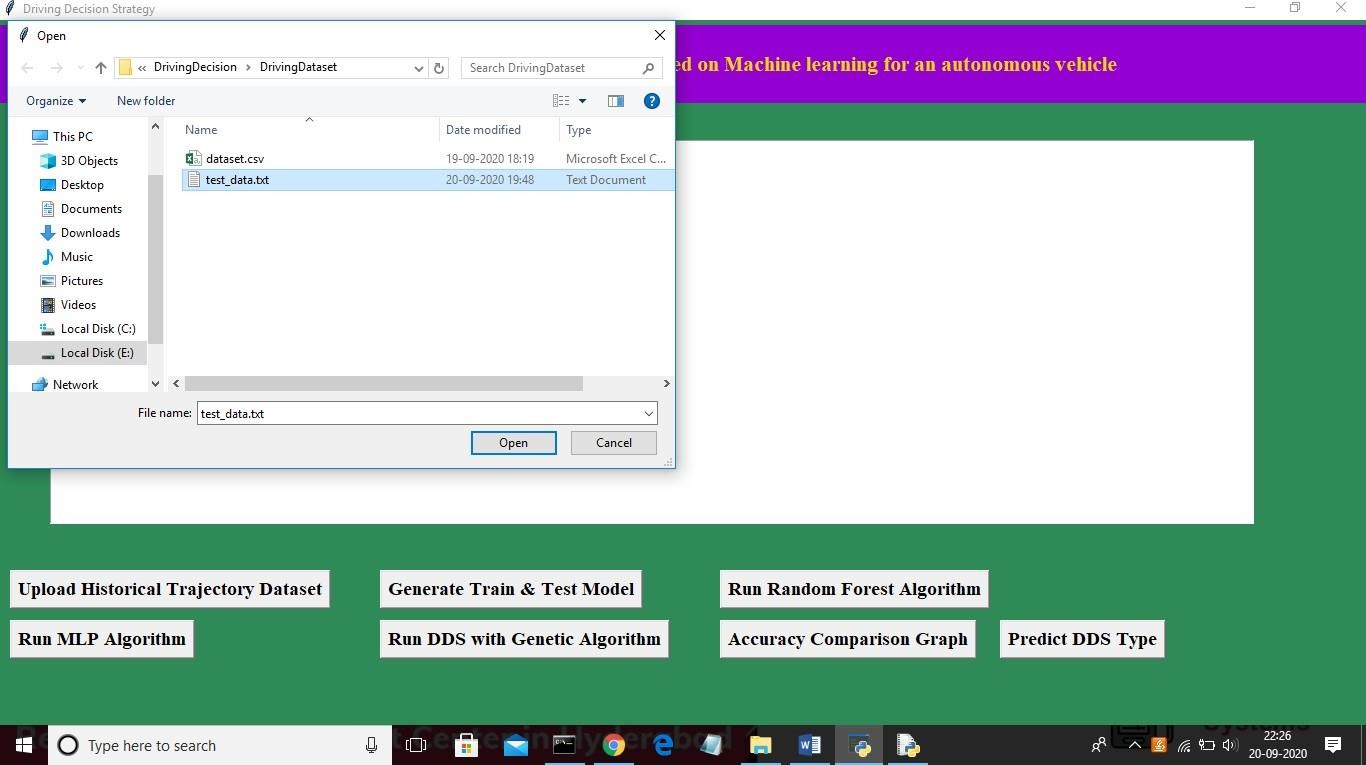
In above black console genetic algorithm starts optimal feature selection



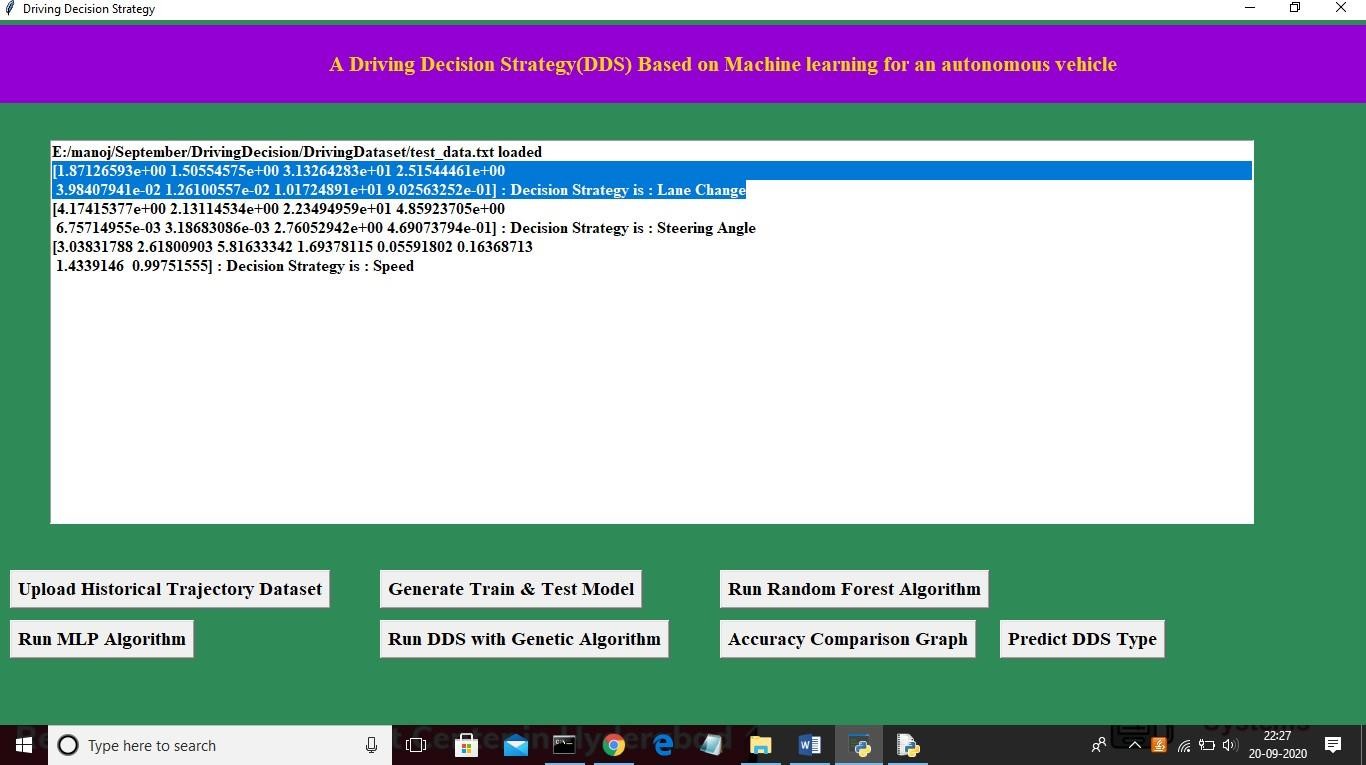
In above screen propose DDS algorithm got 73% prediction accuracy and now click on ‘Accuracy Comparison Graph’ button to get below graph



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



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



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.

# CONCLUSION

This paper proposed a Driving Decision Strategy. It 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.

Because the DDS sends only the key data needed to determine the vehicle's optimal driving strategy to the cloud and analyzes the data through the genetic algorithm, it determines its optimal driving strategy at a faster rate than existing methods. However, the experiments of the DDS were conducted in virtual environments using PCs, and there were not enough resources for visualization.

# FUTURE SCOPE

Future studies should test the DDS by applying it to actual vehicles, and enhance the completeness of visualization components through professional designers.

# 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, Lilian Buzer, 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