**A DRIVING DECISION STRATEGY USING MACHINE LEARNING ALGORITHM FOR AUTONOMOUS VEHICLES**

A Project Report Submitted in partial fulfillment of the requirements  
for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

*Submitted By*

**SISHIR BOHARA**

**(18221A05B9)**

**(18221A05B6)**

**(18221A05C0)**

**(18221A05B8)**

**PAWAN LAMICHHANE**

**RUPALI ADHIKARI**

**DEVRAM YADAV**

Under the Esteemed Guidance of

**Dr. GUNAMANI JENA M.E Ph. D**

 **PROFESSOR**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE  
(AUTONOMOUS)**

(Approved by A.I.C.T.E. New Delhi & Affiliated to J.N.T.U.K. Kakinada)

(Accredited by NAAC with "A" Grade)

ODALAREVU-533210

2018 - 2022

*In memory of*

*Late* ***Mr. Bonam Venkata Chalamayya*** *”Founder Chairman” BVCE*

*Our deepest Gratitude*

**BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

**ODALAREVU-533210**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the project work entitled **“A DRIVING DECISION STRATEGY USING MACHINE LEARNING FOR AUTONOMOUS VEHICLES”** is being submitted for the partial fulfillment of the requirements for the award of the degree of Bachelor Of Technology in **Computer Science and Engineering** to B V C Engineering College, Odalarevu, is a bonafide work done by **SISHIR BOHARA (18221A05B9)**,**PAWAN LAMICHHANE(18221A05B6), RUPALI ADHIKARI(18221A05C0),DEVRAM YADAV(18221A05B8)** under my guidance during the academic year **2018– 2022** and it has been found suitable for acceptance according to the requirement of the University.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree

**Project Guide**

**Head of the Department**

**Dr. GUNAMANI JENA**

**M.E, Ph.D**

**Professor**

**Dr. GUNAMANI JENA**

**M.E, Ph.D**

**Professor**

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We profoundly grateful to express our deep sense of gratitude and respect towards our guide **Dr. GUNAMANI JENA, Associate Professor** of CSE Department, B V C Engineering College, Odalarevu, for her excellent guidance right from selection of the project and her valuable suggestions throughout the project work. Her constant encouragement and support has been the cause of our success, in completing this thesis in the college.

We also grateful to the project Coordinator **Dr. B S N Murthy Professor** CSE Department, B V C Engineering College, Odalarevu, for her excellent guidance right from selection of the project and her valuable suggestions throughout the project work.

We really thankful to **Dr. GUNAMANI JENA Professor** and **Head of the Department of Computer Science and Engineering** for providing the laboratory facilities to the fullest extent as and when required and also for giving us the opportunity to carry out the project work in the college.

We also thankful to **Dr. CHANDRA MOULI V S A, Principal,** B V C Engineering College, Odalarevu, for his support during and till the completion of the project.

We thankful to all the **teaching and non-teaching staff** of Computer Science and Engineering Department, **Management,** B V C Engineering College, Odalarevu, and our **friends** for their direct and indirect help provided to us in completing the project.

Project Associates

**SISHIR BOHARA (18221A05B9)**

**PAWAN LAMICHHANE (18221A05B6)**

**RUPALI ADHIKARI (18221A05C0)**

**DEVRAM YADAV (18221A05B8)**

**ABSTRACT**

The developing technology has made a big stage for several inventions or creations to be formed, worked on, functionalized and used for better improvement of human’s life. The idea of creating an autonomous vehicle is to improve the human driving skills by replacing human’s driving and using artificial intelligence so that there is better usage of safety rules; avoid accidents, proper functioning of roads with well managed traffic and roads.

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 project 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 project compared the DDS with MLP and RF neural network models to validate the DDS.

**TABLE OF CONTENTS**

ABSTRACT………………………………………………………………….…5

TABLE OF CONTENTS………………………………………………….……6

CHAPTER 1………………………………………………………………….....8

INTRODUCTION……………………………………………………………….8

* 1. Objective of the Study……………………………………………………....9
  2. Purpose and Scope…………………………………………………..............9
  3. Existing System…………………………………………………………......11
  4. Proposed System……………………………………………………............11
  5. Machine Learning in Vehicle…………………………………………........12
  6. Motivation………………………………………………………………......12
  7. Problem Statement……………………………………………………….....13
  8. Objective………………………………………………………………........13
  9. Applications…………………………………………………………….......13
  10. Organization of Report……………………………………………........14

CHAPTER 2…………………………………………………………………….15

LITERATURE REWIEW…………………………………………………........15

CHAPTER 3………………………………………………………………..…...18

SYSTEM ANALYSIS…………………………………………………………..18

3.1 Problem Analysis………………………………...........................................18

3.1.1 Autonomous Vehicles………………………………………………....23

3.2 Requirement Specification……………………………………………..…...26

3.2.1 Functional Requirements……………………………………………...28

3.2.2 Non – Functional Requirements………………………………….……28

3.3 System Specification…………………………………..................................28

3.3.2 Hardware and Software Requirements…………………………………...28

CHAPTER 4…………………………………………………….....................30

SYSTEM DESIGN………………................................................................

4.1 System Architecture…………………………………….......................30

4.2 Data Flow Diagram………………………….......................................33

CHAPTER 5…………………………………………………..........................38

IMPLEMENTATION……………………………………………....................38

5.1 Technologies Used……………………………………………...............38

5.2 Implementation and Code……………………………………………….42

5.3 Outcome…………………………………………………………………52

5.4 Extensive Outcome……………………………………………………...60

CHAPTER 6……………………………………………………......................64

TESTING……………………………………………………………...............64

6.1 About Testing…………………………………………………..............64

6.2 Types of Testing……………………………………………………......64

6.2.1 Unit Testing………………………………………………............64

6.2.2 Integration Testing…………………………………………..........65

6.2.3 Acceptance Testing……………………………………………….65

CONCLUSION & FUTURE ENHANCEMENT…….....................................70

REFERENCE…………………………………………………………….…...71

**CHAPTER 1**

**INTRODUCTION**

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].

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 deeprunning operations inside the vehicle, while others use the cloud to compute the vehicle's sensor data. On the other hand, existing studies use only real-time data such as images and sensor data currently 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.

* 1. **Objective of the Study:**

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. **Purpose and Scope:**

**Purpose:**

This project mainly focused on the techniques and measures taken to improve the driving by calculating the data collected by sensors and stored in the cloud. The predictions made by the machine learning algorithms and other algorithms help the drivers to navigate and sense the pedestrians crossing the roads.

Based on the readings and data recorded by the sensors, it has helped the smart vehicle (autonomous vehicle) to be developed in a way that it is basically controlled by artificial intelligence using machine learning algorithms, artificial neural networks. The obtained input from sensor is then passed to genetic algorithm to choose optimal value which helps in faster and efficient prediction.

**Scope:**

The analysis of this project is to show the predictions of how accurate the autonomous vehicle can run within some amount of time with enormous sampling of data. With the rapid growth of the highway transportation system, the number of car ownership has risen year after year which is result in serious traffic conditions [1]. In particular, the incidence of curve accidents and the seriousness of accidents remain high. When the car is turning, there will be a blind zone of sight which is accompanied by increased centrifugal force. The turning radius will decrease and the lateral sliding will occur easily, which is caused collision accidents [2]. In Japan, the traffic accident rate on the curved sections of the road exceeded 41.01% of the total accident rate [3], while the number of traffic accidents on the curved road in China accounted for 7.84% of the total accident. Judging from the severity of the accident, the fatal accidents of the curve occupies 16.3% of all fatal accidents [4]. Other statistics show that the main reasons of accidents in the curved areas are the over-speeding of the turning vehicles during turning, irregularly overtaking lane change and lane occupancy [5]. During driving, many accidents occurred due to driver's inattentiveness or unfamiliarity with the road ahead, especially at the curved road which is the place of the high incidence of accidents [6]. Therefore, if it is possible to detect and recognize the road ahead before the advent of curved road conditions, warn the driver in advance, slowdown and avoid evasion in advance, many unnecessary accidents can be avoided and the safety of life and property can be guaranteed.

* 1. **Existing System:**

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.

**The disadvantages of the existing system are**

* It’s not generating perfect result.
* There is no machine learning algorithms.
  1. **Proposed System:**

To implement this project we have introduced the algorithm called DDS (Driving Decision Strategy) algorithm which is based on Genetic Algorithm to choose optimal gene values which helps in taking better decision or prediction. DDS algorithm, obtained input from sensor and then pass to genetic algorithm to choose optimal value which helps in faster and efficient prediction.

**The advantages of the proposed system are**

1. Propose DDS with genetic algorithm performance is comparing with existing machine learning algorithm such as Random Forest and MLP (multilayer perceptron algorithm.).
2. Propose DDS shows better prediction accuracy compare to random forest and MLP.
   1. **Machine Learning in Vehicle:**

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.

* 1. **Motivation:**

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, existing studies use only real-time data such as images and sensor data currently collected from vehicles to determine how the vehicle is driving.

* 1. **Problem Statement:**

In this project we are describing concept for driving decision strategy by observing vehicle internal data such as steering and RPM level to predict various classes such as speed (steering), changing lane etc. All existing techniques were concentrated on external data such as road condition and pedestrians etc. But not concentrated on internal values. So to take efficient determination of steering condition and changing lane author is analysing internal data.

All internal data will be collected from sensor and then store on cloud and then application will read data from cloud and then apply machine learning algorithms to determine or predict steering condition or changing lane.

* 1. **Objective**

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. **Application:**

Self-driving cars.

* 1. **Organization Report:**

The rest of the report is organized into 5 chapters. After this introductory chapter, the next chapter-2 describes about the survey of the existing system. This establishes a context of the research conducted by the researchers up until now in the field of driving decision strategy using ML and **Genetic algorithm**.

Chapter-3 describes the proposed system. This starts with the introduction of the dataset, the models that have been used in the report. Then it covers the architecture of the proposed system. Describes the process and the algorithms used, the details of the software used for the research work. It also describes the evaluation parameters used for this study.

Chapter-4 shows the experiment and the results. It appears the confusion network of each model and the comparison graph. This helps us to identify which model is the most efficient for the stock market trend prediction using ML and DL algorithms. .

Chapter-5 gives a conclusion about the result of all the models in this research paper and gives suggestions about which model to use when. It gives a new direction of future

**CHAPTER 2**

**LITERATURE REVIEW**

**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 project 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.: 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), Flex Ray, 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 Flex Ray 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.

**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.**

This paper presents a method for segmenting a 3D point cloud into planar surfaces using recently obtained discrete-geometry results. In discrete geometry, a discrete plane is defined as a set of grid points lying between two parallel planes with a small distance, called thickness. In contrast to the continuous case, there exist a finite number of local geometric patterns (LGPs) appearing on discrete planes. Moreover, such an LGP does not possess the unique normal vector but a set of normal vectors. By using those LGP properties, we first reject non- linear points from a point cloud, and then classify non-rejected points whose LGPs have common normal vectors into a planar-surface-point set. From each segmented point set, we also estimate the values of parameters of a discrete plane by minimizing its thickness.

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

In Intelligent Transportation Systems (ITS), logistics distribution and mobile e-commerce, the real-time, accurate and reliable vehicle trajectory prediction has significant application value. 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 (DHMTP) 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, especially when traffic flow is greater, such as this time from weekday morning to evening. Moreover, the time performance of DHMTP algorithm is also clearly improved compared with TPMO algorithm.

**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**

Real-time and reliable measurements of the effluent quality are essential to improve operating efficiency and reduce energy consumption for the wastewater treatment process. Due to the low accuracy and unstable performance of the traditional effluent quality measurements, we propose a selective ensemble extreme learning machine modeling method to enhance the effluent quality predictions. Extreme learning machine algorithm is inserted into a selective ensemble frame as the component model since it runs much faster and provides better generalization performance than other popular learning algorithms. Ensemble extreme learning machine models overcome variations in different trials of simulations for single model. Selective ensemble based on genetic algorithm is used to further exclude some bad components from all the available ensembles in order to reduce the computation complexity and improve the generalization performance. The proposed method is verified with the data from an industrial wastewater treatment plant, located in Shenyang, China. Experimental results show that the proposed method has relatively stronger generalization and higher accuracy than partial least square, neural network partial least square, single extreme learning machine and ensemble extreme learning machine mode

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 Problem Analysis:**

Problem analysis is an important activity that takes place when we are building a new system or changing the existing one. Analysis helps to understand the existing system and the requirements necessary for building the new system. If there is no existing system then analysis defines the only requirements. One of the important factors in the system analysis is to understand the system and its problems. A good understanding of the system enables designer to identify and correct problems. Based on the drawbacks of the existing system the system is being planned, so the total definition of the problem has been analyzed.

1. **Algorithm:**

In this study, we use three most popular machine learning methods (**Random forest**, **MLP, Genetic algorithm)**

**1.1 Random Forest Algorithm:**

This model has three random concepts, randomly choosing training data when making trees, selecting some subsets of features when splitting nodes and considering only a subset of all features for splitting each node in each simple decision tree. During training data in a random forest, each tree learns from a random sample of the data points.

Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The -forest II it builds, is an ensemble of Decision Trees, most of the time trained with the -bagging ll method. The general idea of the bagging method is that a combination of learning models increases the overall result. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. We will talk about random forest in classification, since classification is sometimes considered the building block of machine learning.

A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms.

A random forest is a machine learning technique that’s used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

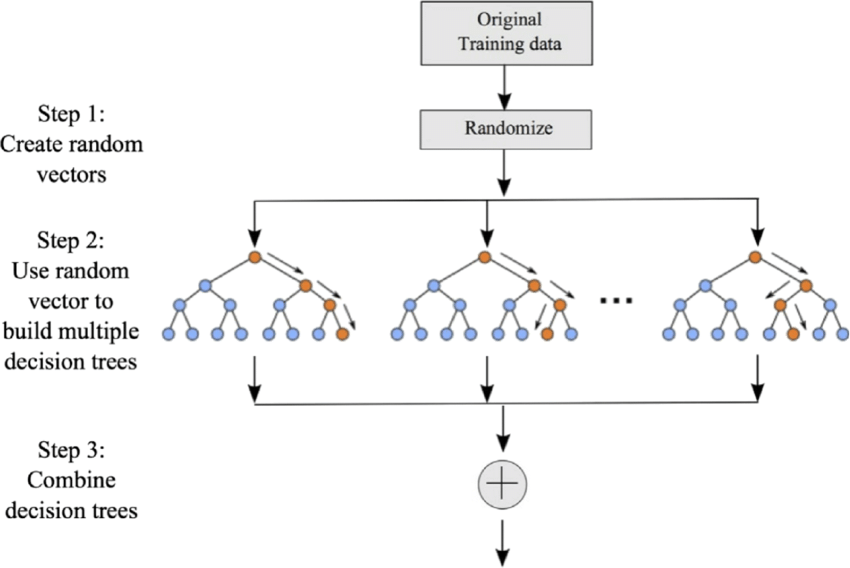


Fig.1 Random forest

A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like scikit-learn).

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.

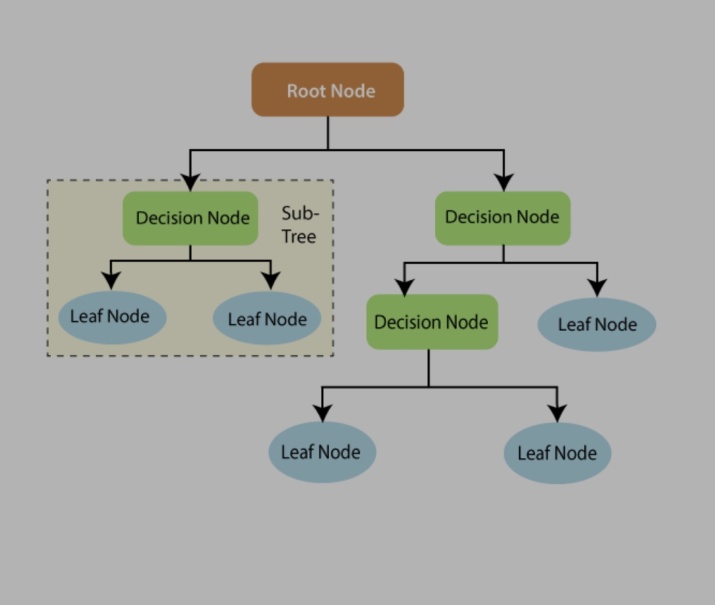


Fig.2 Decision tree with nodes

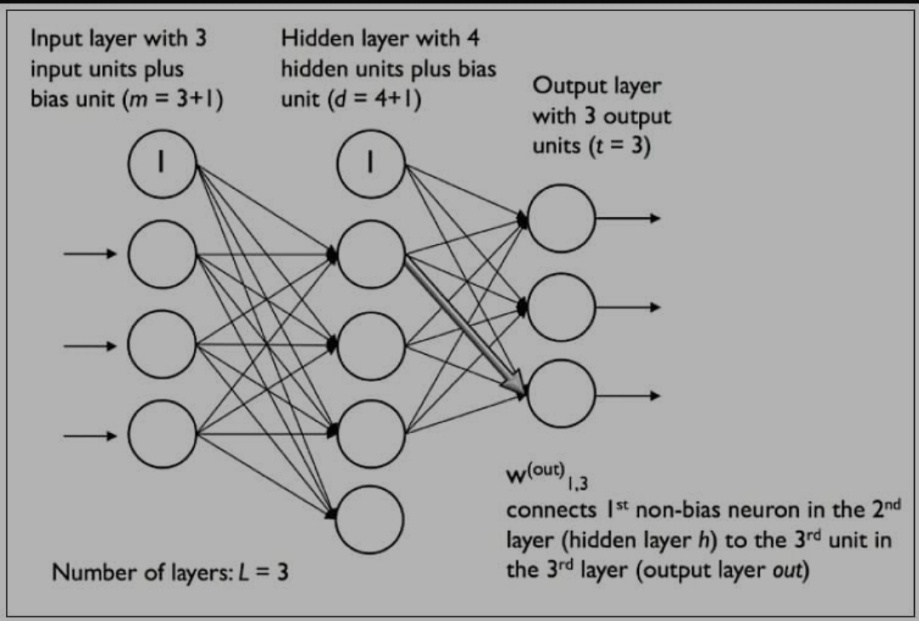
* 1. **MLP(Multi-layer Perceptron):**

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. MLP utilizes a supervised learning technique called back propagation for training.

A multilayer artificial neuron network is an integral part of deep learning. And this lesson will help you with an overview of multilayer ANN along with over-fitting and under-fitting fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).

The number of layers and the number of neurons are referred to as hyper parameters of a neural network, and these need tuning. Cross-validation techniques must be used to find ideal values for these.

The weight adjustment training is done via back propagation. Deeper neural networks are better at processing data. However, deeper layers can lead to vanishing gradient problems. Special algorithms are required to solve this issue.

1. It has 3 layers including one hidden layer. If it has more than 1 hidden layer, it is called a deep ANN. An MLP is a typical example of a feed forward artificial neural network.
2. 
3. Fig.3 MLP with activation function

A multi-layered perceptron consists of interconnected neurons transferring information to each other, much like the human brain. Each neuron is assigned a value. The network can be divided into three main layers.

**Input Layer**

This is the initial layer of the network which takes in an input which will be used to produce an output.

**Hidden Layer(s)**

The network needs to have at least one hidden layer. The hidden layer(s) perform computations and operations on the input data to produce something meaningful.

**Output Layer**

The neurons in this layer display a meaningful output.

* 1. **Genetic algorithm**

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to be integer-valued.

* A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature."
* It is used to solve optimization problems in machine learning. It is one of the important algorithms as it helps solve complex problems that would take a long time to solve.
* Genetic Algorithms are being widely used in different real-world applications, for example, Designing electronic circuits, code-breaking, image processing, and artificial creativity.

(SGA \_\_ simple genetic algorithm)

**function sga ()**{

Initialize sample data;

Calculate fitness function;

While (fitness value != termination criteria) {

Selection;

Crossover;

Mutation;

Calculate fitness function;

}

}

It is a contentious point whether GA’s can be applied to machine learning. The point has been explored and explained in the following work by taking example of chess playing. The definition and types of classifier systems have been explained in the first section followed by explanation of machine learning. This is followed by the brief analysis of genetic algorithms. The application of GA’s to machine learning taking the example of chess has been explained in section IV. It has been found that if there are many rules to be applied for a particular condition then GA’s give an effective solution if the rules can be assigned correct fitness values.

**2. Autonomous Vehicle Operational Models**

**Introduction:**

The future is ultimately unknowable but planning requires predicting impending conditions and needs. Many decision-makers and practitioners (planners, engineers and analysts) wonder how autonomous (also called self-driving or robotic) vehicles will affect future travel demands, and therefore the need for roads, parking facilities and public transit services, and what public policies can minimize their risks and maximize their benefits (APA 2016; Berrada and Leurent 2017; Grush and Niles 2018; Guerra 2015; Kockelman and Boyles 2018; Larco 2022; Milakis, van Arem and van Wee 2017; Shaheen, Totte and Stocker 2018; Sperling 2017).



Fig.4 How does the working mechanism looks like



Fig.5 Display in front

There is considerable uncertainty about these issues. Optimists predict, based on experience with previous technological innovations such as digital cameras, smart phones and personal computers, that autonomous vehicles will soon be sufficiently reliable and affordable to replace most human driving, providing huge savings and benefits (Johnston and Walker 2017; Keeney 2017; Kok, et al. 2017). However, there are good reasons to be skeptical of such class.

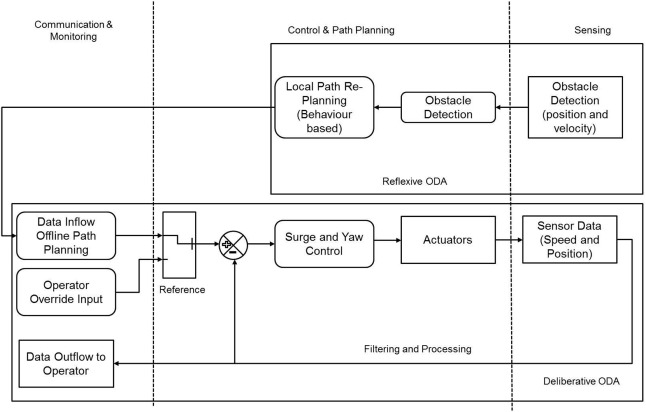


Fig.6 Working of different parts in the system

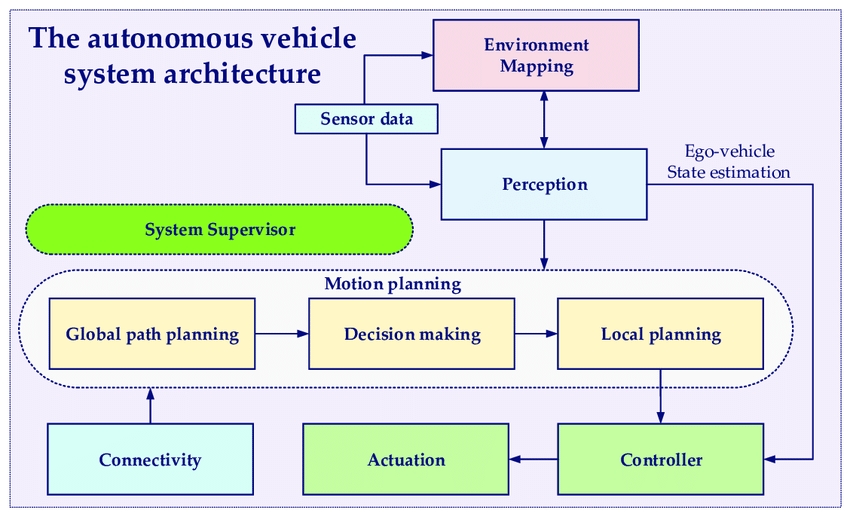
****

Fig.7 Systematic workflow of autonomous vehicle

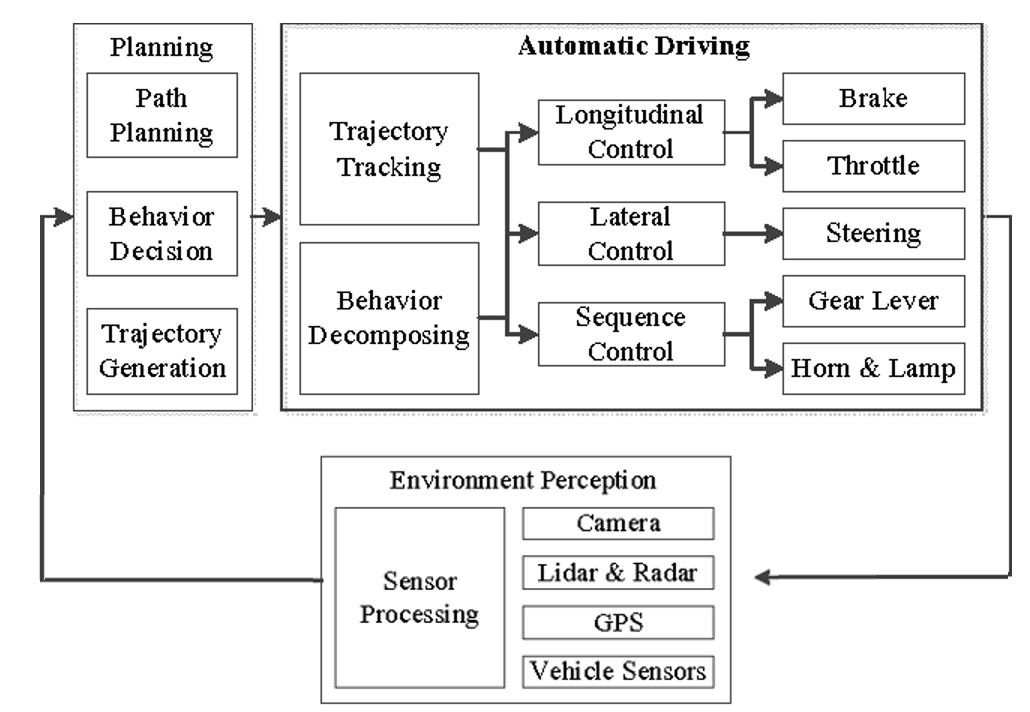


Fig.8 Systematic behavior of automated vehicle after DDS implementation

**3.3 Specification Requirement:**

A **Software Requirements Specification (SRS) -** A requirements specification for a software system - is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non­functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

**3.3.1 System requirements specification:** A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stake holders to help identify business problems and propose solutions. Within the systems development life cycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* **Business requirements** describe in business terms what must be delivered or accomplished to provide value.
* **Product requirements** describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development style.

A systems engineering, a requirement can be a description of what a system must do, referred to as a functional requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-fimctional requirements, or 'performance requirements' or 'quality of service requirements. Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying how the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, implementation constraints on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products. In Software Engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

* + 1. Functional Requirements:

Functional requirements are very important system requirements in the system design process. These requirements are the technical specifications, system design parameters and guidelines, data manipulation, data processing, and calculation modules etc, of the proposed system. Functional requirements are in contrast to Non-Functional requirements which are descripitive of the parameters of system performance, quality attributes, reliability and security, cost, constraints in design/implementation, etc

* + 1. Non- Functional Requirements:

Non-functional requirements tend to be stated in terms of constraints on the results of tasks which are given as functional requirements (e.g. constraints on the speed or efficiency of a given task), a task based functional requirements statement is a useful skeleton upon which to construct a complete requirements statement. That is the approach taken in this work. It can be helpful to think of non-functional requirements as adverbially related to tasks or functional requirements.

Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are –constraints II, quality attributes II, quality goals II, -quality of service requirements II and non-behavioral requirements II.

**3.4 Software Specification:**

HARDWARE REQUIREMENTS:

System : Pentium i5

Hard Disk : 500 GB.

Monitor : 15’’ LED

Input Devices : Keyboard, Mouse

RAM : 4 GB

SOFTWARE REQUIREMENTS**:**

Operating system : Windows 10.

Coding Language : Python 3.7.0

Python Packages : NumPY, Pandas, Matplotlib,Keras

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Architecture/Framework**:

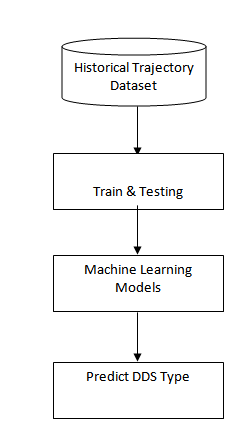


Fig.9 System Architecture

**4.2 Algorithm and Process Design:**

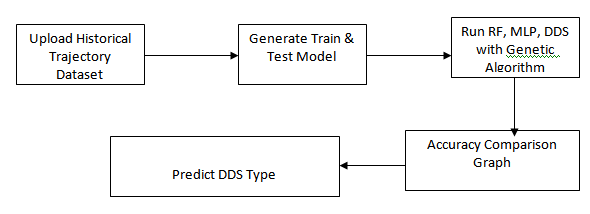
****

Fig.10 Algorithm and Process Design

**Data collection**

To implement this project we are using historical vehicle trajectory dataset as we don’t have sensors to collect data so we are using trajectory dataset. In dataset if user is slowing down vehicle then it has some sensor value with class label as ‘lane changing’. Similarly based on values we have different classes in dataset. Machine learning algorithm will be trained on such dataset and then when we apply test data on trained model then algorithm will predict class for that test data.

Below are the dataset details ‘DrivingDataset’

**trajectory\_id,start\_time,end\_time,rpm\_average,rpm\_medium,rpm\_max,rpm\_std,speed\_average,speed\_medium,speed\_max,speed\_std,labels**

20071010152332,2007-10-10T15:23:32.000000000,2007-10-10T15:32:59.000000000,2.21513818073,2.27421615004,2.85853043655,0.428624902772,-0.005093147516729999,-0.00230819670622,0.0647143832211,0.0377402391782,speed

20071011011520,2007-10-11T01:15:20.000000000,2007-10-11T01:22:10.000000000,3.71181007816,3.65065107266,6.35783373513,1.9271696164900003,-0.016218030061,-0.00147783417456,0.104789889519,0.09341315155410003,speed

20080628053717,2008-06-28T05:37:17.000000000,2008-06-28T05:46:42.000000000,4.65889245882,3.12829931751,13.0268086376,4.09914234541,0.00404703387141,0.0124246102197,2.11899984839,0.7521915347560001,steering\_angle

20080628124807,2008-06-28T12:48:07.000000000,2008-06-28T12:57:16.000000000,1.71674094314,1.31398945454,18.5776836518,2.18497323244,-0.0312684175217,0.0308633583269,2.93888558793,0.7139256777420001,steering\_angle

20080825044741,2008-08-25T04:47:41.000000000,2008-08-25T05:05:12.000000000,2.38238360506,1.5371758264500002,20.919113327999998,2.865359735,-0.00720368601786,-0.000910857743471,2.01833073218,0.471527016571,lane\_change

In above dataset all bold names are the dataset column names and below it are the dataset values. In dataset we can see sensor report each record with trajectory id, date, time and with speed and rpm details. In last column we can see labels as LANE\_CHANGE, STEERING ANGLE and SPEED and with above dataset values and with label we will train machine learning algorithm and calculate accuracy.

Below are the test data which will not have any class label and it will have only sensor values and by applying sensor values on trained model we can predict or determine driving decision.

**trajectory\_id,start\_time,end\_time,rpm\_average,rpm\_medium,rpm\_max,rpm\_std,speed\_average,speed\_medium,speed\_max,speed\_std**

20080823105259,2008-08-23T10:52:59.000000000,2008-08-23T11:03:41.000000000,1.871265931,1.50554575041,31.326428333800006,2.51544461011,0.039840794139,0.0126100556557,10.1724891367,0.90256325184

20080821073812,2008-08-21T07:38:12.000000000,2008-08-21T08:30:53.000000000,4.17415377139,2.13114534045,22.3494958748,4.85923705089,0.00675714954958,0.003186830858360001,2.76052942367,0.469073794101

20080913092418,2008-09-13T09:24:18.000000000,2008-09-13T09:24:36.000000000,3.03831788365,2.6180090273700003,5.81633341636,1.6937811468,0.0559180233599,0.163687128621,1.43391460095,0.997515549234

In above test data we can see only test values are there but not class label and after applying above test data on machine learning trained model we can predict/determine driving strategy such as going on speed, changing lane or steering angle.

**Upload historical trajectory Dataset**: using this module we will upload dataset to application and then find out total number records.

**Generate train & test model:** This module is read dataset and to split dataset into train and test part to generate machine learning train model

**Run Random Forest:** Using this module we will split dataset into train and test and then build Random Forest trained model. Trained model will be applied on test data to calculate and test prediction accuracy

**Run MLP Algorithm**: Using this module we will split dataset into train and test and then build MLP trained model. Trained model will be applied on test data to calculate and test prediction accuracy.

* 1. Data Flow Diagram:

Data flow diagrams provide a graphical representation of how information moves between processes in a system. Data flow diagrams follow a hierarchy; that is, a diagram may consist of several layers, each unique to a specific process or data function.

Symbols Used in DFD’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | | | **Meaning** |
| **Process** | | | **Single process: x A. Circle is used to represent the entire system.** |
|  | | | **Data flow: ±An arrow is used to represent the flow of data between the process and external entities.** |
|  |  | **External**  **entity** | **External entity: A square or rectangle represents any person or organization that sends data to or receives data from**  **the system.** |
|  | **Data store** | | **Data store: An open rectangle represents**  **the location where data is stored. It could be a filing cabinet, hard disk.** |

Fig.11 DFD Diagram

**LEVEL 0 and 1 DIAGRAM:\**

**Import**

**Process**

**Visualize**

**Model**

**Evaluate**

Fig.12 Machine Learning Workflow Diagram

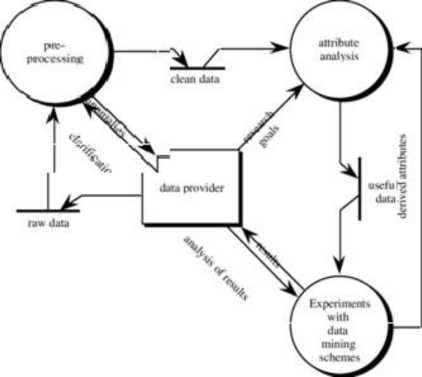


Fig.13 Level 1 DFD Diagram

**Use-case Diagram:**

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

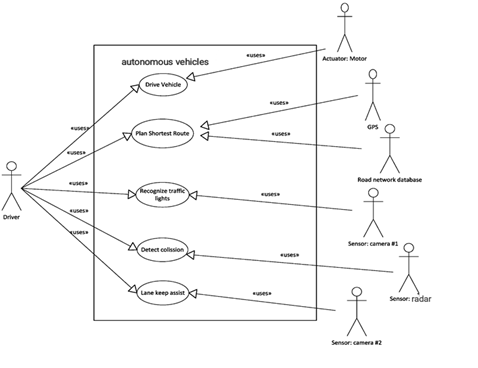


Fig.14 Use-case diagram

Sequence Diagram:

A sequence diagram shows object interactions arranged in time sequence.It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event.

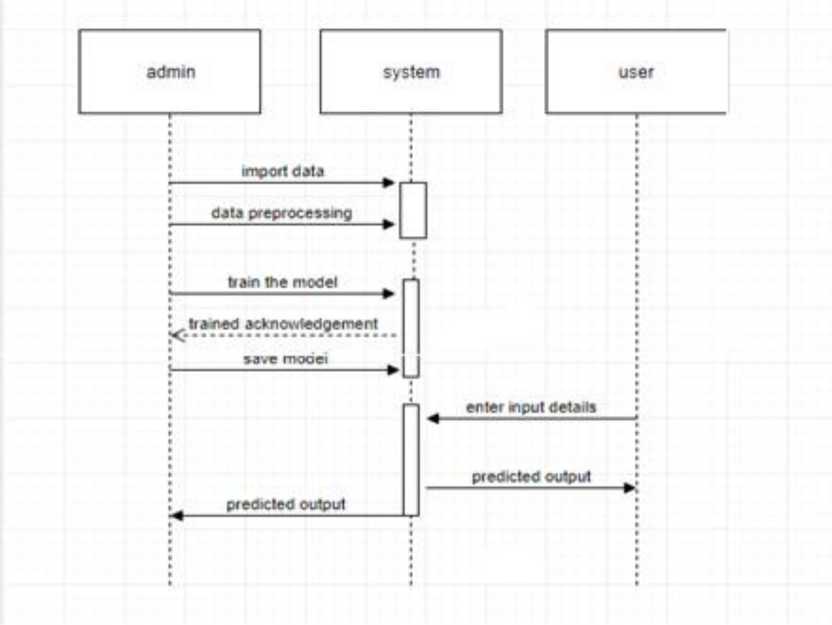


Fig.15 Sequence diagram

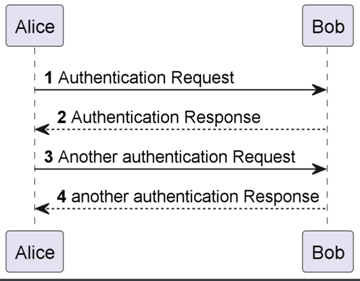
****

Fig.16 Sequence Diagram

**Class diagram:**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of objectoriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram

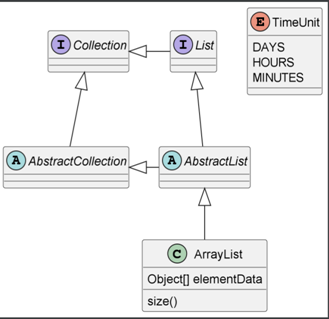
****

Fig.17 Class diagram

**CHAPTER 5**

**IMPLEMENTATION AND OUTCOMES**

* 1. Technologies Used:

**PYTHON:**

Python is a popular platform used for research and development of production systems. It is a vast language with number of modules, packages and libraries that provides multiple ways of achieving a task.

Python and its libraries like NumPy, SciPy, Scikit-Learn, and Matplotlib are used in data science and data analysis. They are also extensively used for creating scalable machine learning algorithms. Python implements popular machine learning techniques such as Classification, Regression, Recommendation, and Clustering.

Libraries and Packages:

To understand machine learning, you need to have basic knowledge of Python programming. In addition, there are a number of libraries and packages generally used in performing various machine learning tasks as listed below:

* Numpy - is used for its N-dimensional array objects
* Pandas - is a data analysis library that includes data frames
* Matplotlib - is 2D plotting library for creating graphs and plots
* Scikit-learn - the algorithms used for data analysis and data mining tasks
* Keras**:** Keras is a deep learning API written in Python, running on top of the machine learning platform Tensor Flow. It was developed with a focus on enabling fast experimentation

Installation Steps Involved in Machine Learning:

A machine learning project involves the following steps:

* Defining a Problem
* Preparing Data
* Evaluating Algorithms
* Improving Results
* Presenting Results

The best way to get started using Python for machine learning is to work through a project end-to-end and cover the key steps like loading data, summarizing data, evaluating algorithms and making some predictions. This gives you a replicable method that can be used dataset after dataset. You can also add further data and improve the results.

Installation:

You can install software for machine learning in any of the two methods as discussed here:

Method 1:

Download and install Python separately from python.org on various operating systems as explained below:

To install Python after downloading, double click the .exe (for Windows) or .pkg (for Mac) file and follow the instructions on the screen.

For Linux OS, check if Python is already installed by using the following command at the prompt:

$ python --version....

If Python 3.7 or later is not installed, install Python with the distribution's package manager. Note that the command and package name varies.

On Debian derivatives such as Ubuntu, you can use apt:

$ sudo apt-get install python3.7.0

Now, open the command prompt and run the following command to verify that Python is installed correctly:

$ python3 —version

Python 3.7.0

Similarly, we can download and install necessary libraries like numpy, matplotlib etc. individually using installers like pip. For this purpose, you can use the commands shown here:

$pip install numpy

$pip install matplotlib

$pip install pandas

Method 2:

Alternatively, to install Python and other scientific computing and machine learning packages simultaneously, we should install Anaconda distribution. It is a Python implementation for Linux, Windows and OSX, and comprises various machine learning packages like numpy, scikit-learn, and matplotlib. It also includes Jupyter Notebook, an interactive Python environment. We can install Python 2.7 or any 3.x version as per our requirement.

To download the free Anaconda Python distribution from Continuum Analytics, you can do the following:

Visit the official site of Continuum Analytics and its download page. Note that the installation process may take 15-20 minutes as the installer contains Python, associated packages, a code editor, and some other files. Depending on your operating system, choose the installation process as explained here:

**For Windows:**

Select the Anaconda for Windows section and look in the column with Python 2.7 or 3.7.0. You can find that there are two versions of the installer, one for 32-bit Windows, and one for 64-bit Windows. Choose the relevant one.

**For Mac OS:** Scroll to the Anaconda for OS X section. Look in the column with Python 2.7 or 3.7.0. Note that here there is only one version of the installer: the 64-bit version.

**For Linux OS:** We select the "Anaconda for Linux" section. Look in the column with Python 2.7 or 3.x.

Note that you have to ensure that Anaconda‘s Python distribution installs into a single directory, and does not affect other Python installations, if any, on your system.

To work with graphs and plots, we will need these Python library packages: **matplotlib.**If you are using Anaconda Python, your system already has numpy, matplotlib, pandas, seaborn, etc. installed. We start the Anaconda Navigator to access either **Jupyter Note book** or Spyder **IDE** of python.

After opening either of them, type the following commands.

import numpy

import matplotlib

Now, we need to check if installation is successful. For this, go to the command line and type in the following command:

$ python

Python 3.7.0 (Anaconda custom (64-bit)

Next, you can import the required libraries and print their versions as shown:

» import numpy

» print ( numpy.\_version\_1.14.2)

» import matplotlib

» print ( matplotlib.\_version\_2.1.2)

» import pandas

» print (pandas.\_version\_0.22.0)

**5.2 Implementation and Outcome**

**About the data:**

To implement this project we are using historical vehicle trajectory dataset as we don’t have sensors to collect data so we are using trajectory dataset. In dataset if user is slowing down vehicle then it has some sensor value with class label as ‘lane changing’. Similarly based on values we have different classes in dataset. Machine learning algorithm will be trained on such dataset and then when we apply test data on trained model then algorithm will predict class for that test data

**Evaluation Metrics**:

F1-Score, Accuracy and Receiver Operating Characteristics-Area Under the Curve (ROC-AUC) metrics are employed to evaluate the performance of our models. For Computing F1-score and Accuracy, Precision and Recall must be evaluated by

* FPR=False Positive Rate
* TPR=True Positive Rate
* Accuracy
* Precision
* Recall
* F1-score

For this, the calculation of values is measured based on:

• True positive (TP) = No. of events, correctly determined.

• False negative (FN) = No. of events, inaccurately anticipated and not required.

• False-positive (FP) = No. of events, incorrectly predicted.

• True negative (TN) = No. of events, correctly anticipated and not required.

**False Positive Rate (FPR):** Itis a metric that can be used to assess [machine learning accuracy.](https://deepchecks.com/glossary/machine-learning-model-accuracy/) It is defined as:

**FPR=FP/(FP+TN)**

**True Positive Rate (TPR):** Itis a synonym for recall and is therefore defined as

**TPR=FP/(FP+TN)**

**Accuracy:** It is the most important performance measure and it is easily done by a ratio of correctly predicted observations to the total observations.

**Accuracy=(TN+TP)/(TP+FP+TN+FN)**

**Recall:** It is the ratio which correctly predicts positive observations among all observations in original data.

**Recall= TP/(TP+FN)**

**Precision:** It is used to calculate the correctly identified values. This means to calculate the total number of software’s which are correctly predicted as positive from the total number of software’s predicted positive. It is defined as

**Precision = TP/ (TP + FP)**

**F1-score:** The F-score is a way of combining the [precision and recall](https://deepai.org/machine-learning-glossary-and-terms/precision-and-recall) of the model, and it is defined as the mean of the model’s precision and recall. It is also called as F-score. It is defined as

**F1 Score = 2(Precision Recall/Precision + Recall)**

ROC-AUC is another powerful metric for classification problems, and is calculated based on the area under ROC-AUC curve from prediction scores.

**5.2 Code Implementation:**

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 xgboost import XGBClassifier

from genetic\_selection import GeneticSelectionCV

global main, text

main = tkinter.Tk()

main.title("Driving Decision Strategy") #designing main screen

main.geometry("1300x1200")

global filename

global X, Y

le = LabelEncoder()

global mlp\_acc, rf\_acc, dds\_acc, extension\_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, X, Y

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\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\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=2, #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\n")

for i in range(0,3):

y\_test[i] = 5

classifier = selector

def runExtension():

global extension\_acc

dds = XGBClassifier()

selector = GeneticSelectionCV(dds, #algorithm name

cv=5,

verbose=1,

scoring="accuracy",

max\_features=5,

n\_population=2, #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, Y)

text.insert(END,"Extension 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

extension\_acc = accuracy\_score(y\_test,prediction\_data)\*100

text.insert(END,"Extension DDS with XGBoost Precision : "+str(dds\_precision)+"\n")

text.insert(END,"Extension DDS with XGBoost Recall : "+str(dds\_recall)+"\n")

text.insert(END,"Extension DDS with XGBoost FMeasure : "+str(dds\_fmeasure)+"\n")

text.insert(END,"Extension DDS with XGBoost Accuracy : "+str(extension\_acc)+"\n")

def graph():

height = [rf\_acc, mlp\_acc,dds\_acc, extension\_acc]

bars = ('Random Forest Accuracy','MLP Accuracy','DDS with Genetic Algorithm Accuracy','DDS with XGBoost Extension')

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")

def mainMethod():

global main, text

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', 13, '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 & Test Model", command=generateTrainTestData)

trainButton.place(x=350,y=550)

trainButton.config(font=font1)

rfButton = Button(main, text="Run Random Forest Algorithm", command=runRandomForest)

rfButton.place(x=660,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=350,y=600)

ddsButton.config(font=font1)

extensionButton = Button(main, text="Extension DDS with Genetic & XGBoost", command=runExtension)

extensionButton.place(x=660,y=600)

extensionButton.config(font=font1)

graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)

graphButton.place(x=1000,y=600)

graphButton.config(font=font1)

predictButton = Button(main, text="Predict DDS Type", command=predictType)

predictButton.place(x=10,y=650)

predictButton.config(font=font1)

main.config(bg='sea green')

main.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

mainMethod()

**5.3 Outcome:**

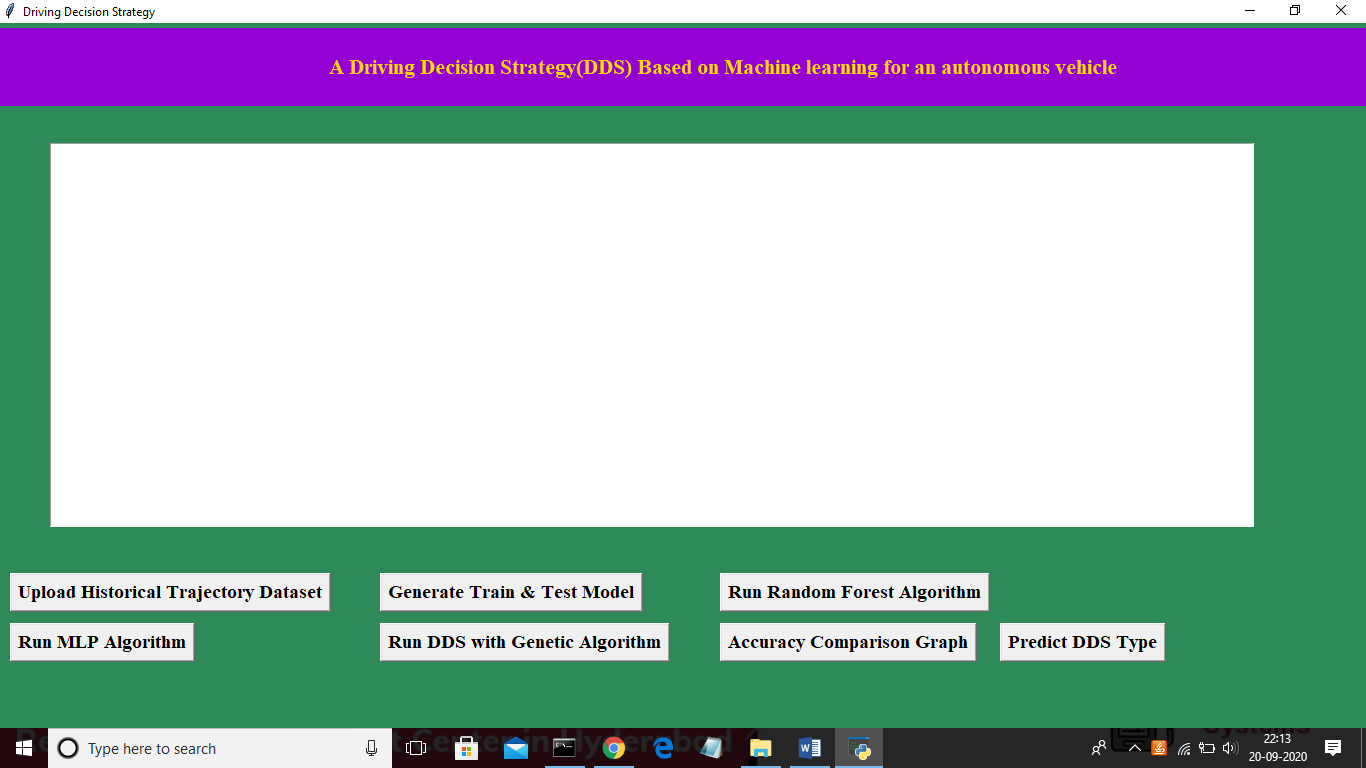


Fig.18 Front screen

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

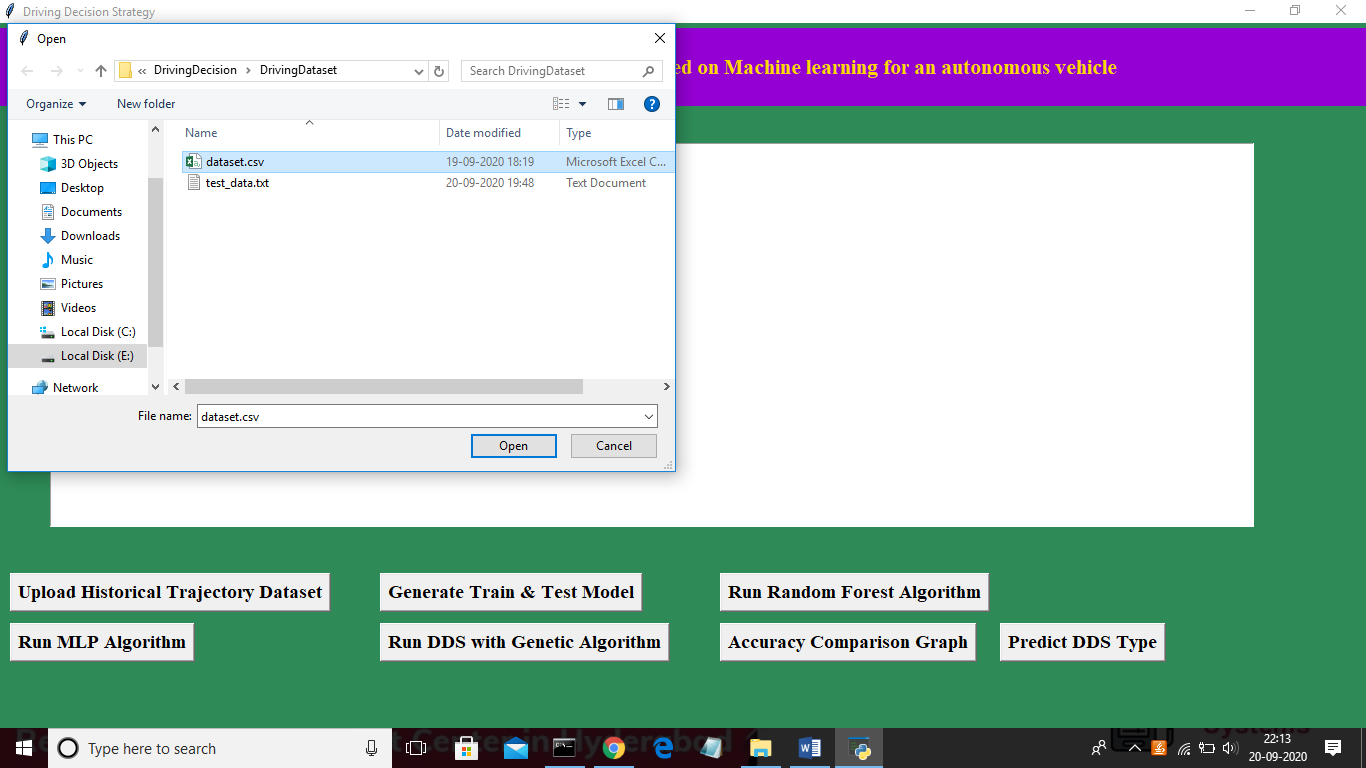


Fig.19 Importing dataset

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

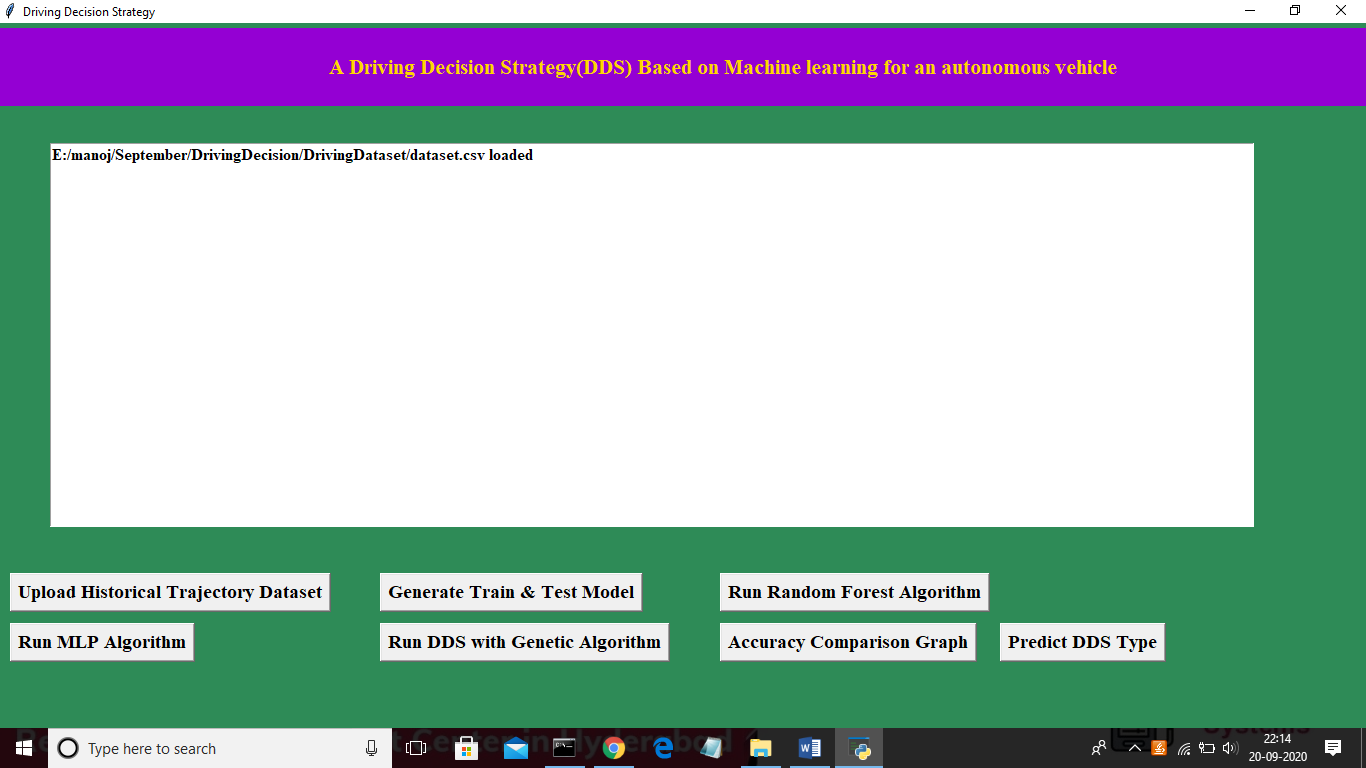


Fig.20

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.

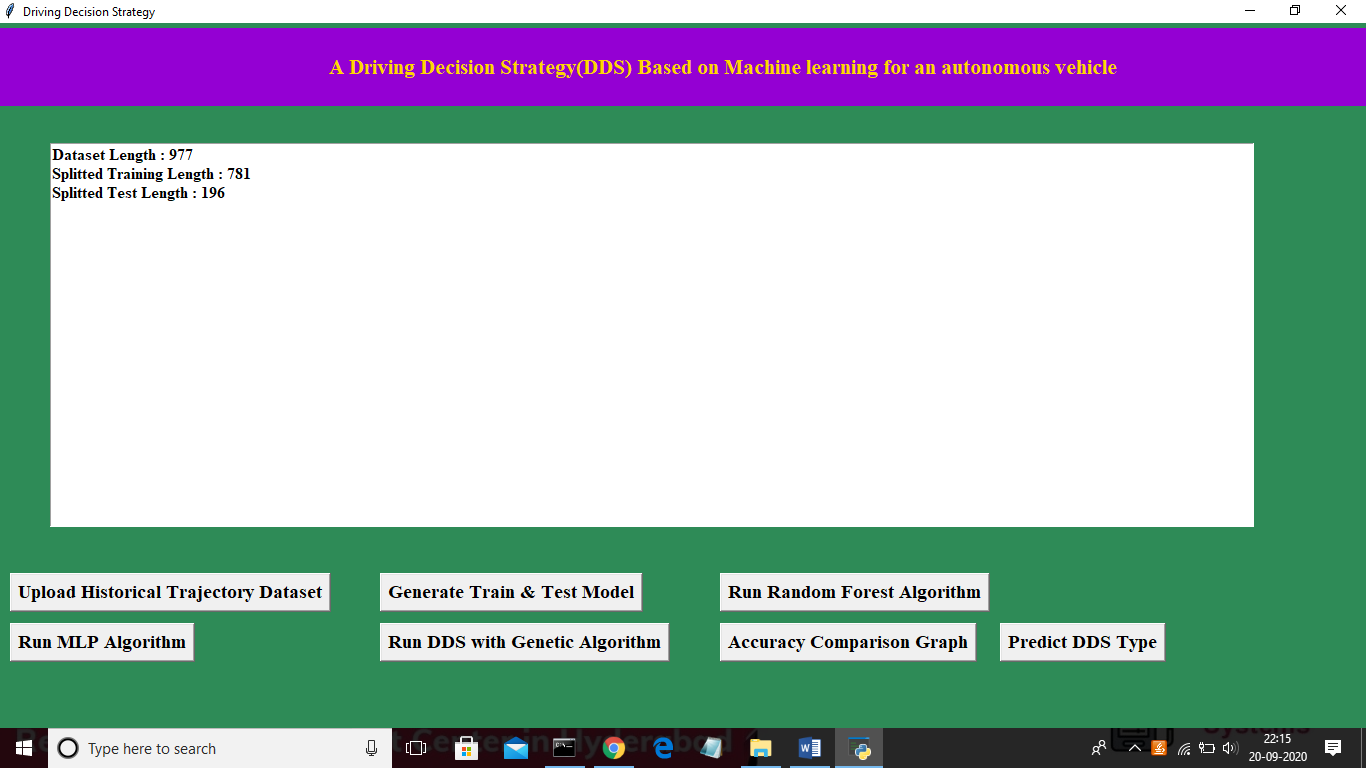


Fig.21 Generating Train and Test Model

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

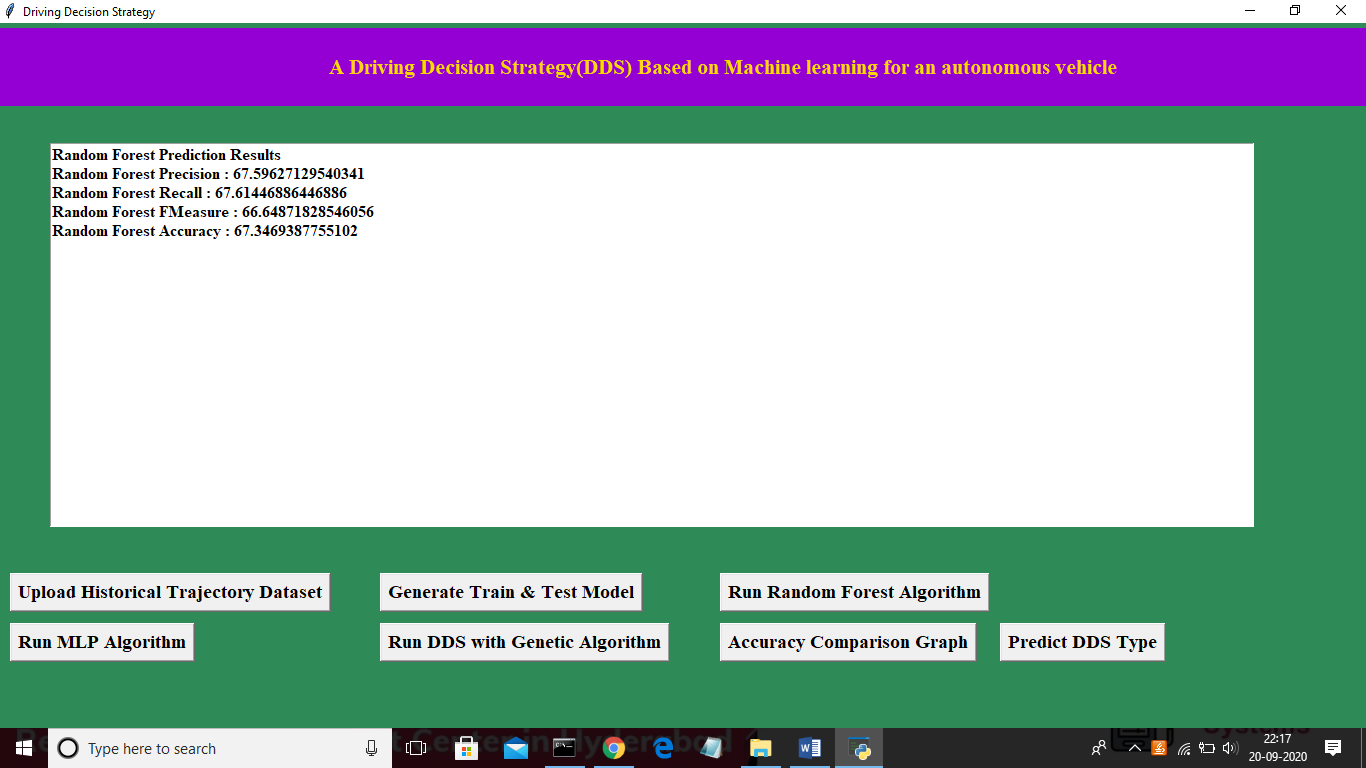


Fig.22 Running Random Forest Algorithm

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

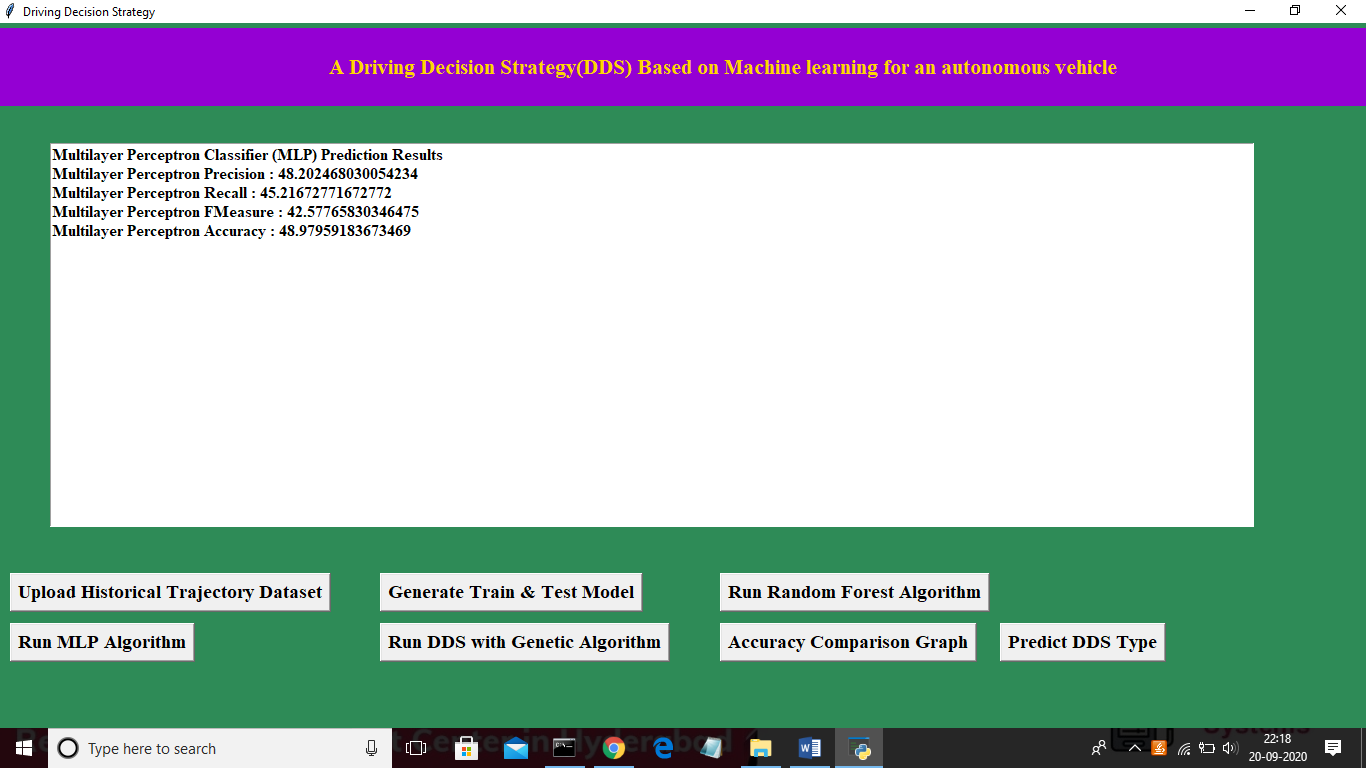


Fig.23 Running MLP Algorithm

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

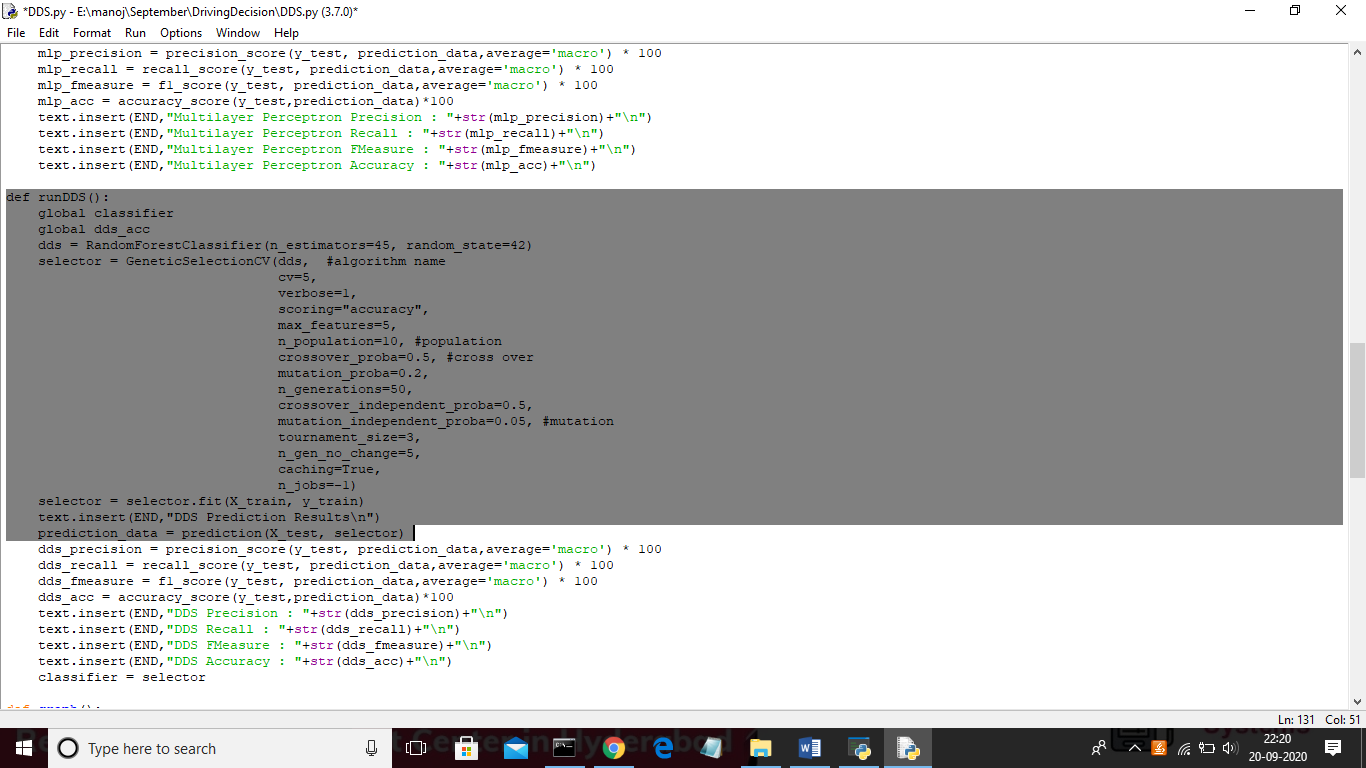


Fig.24 Coding implementation

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.

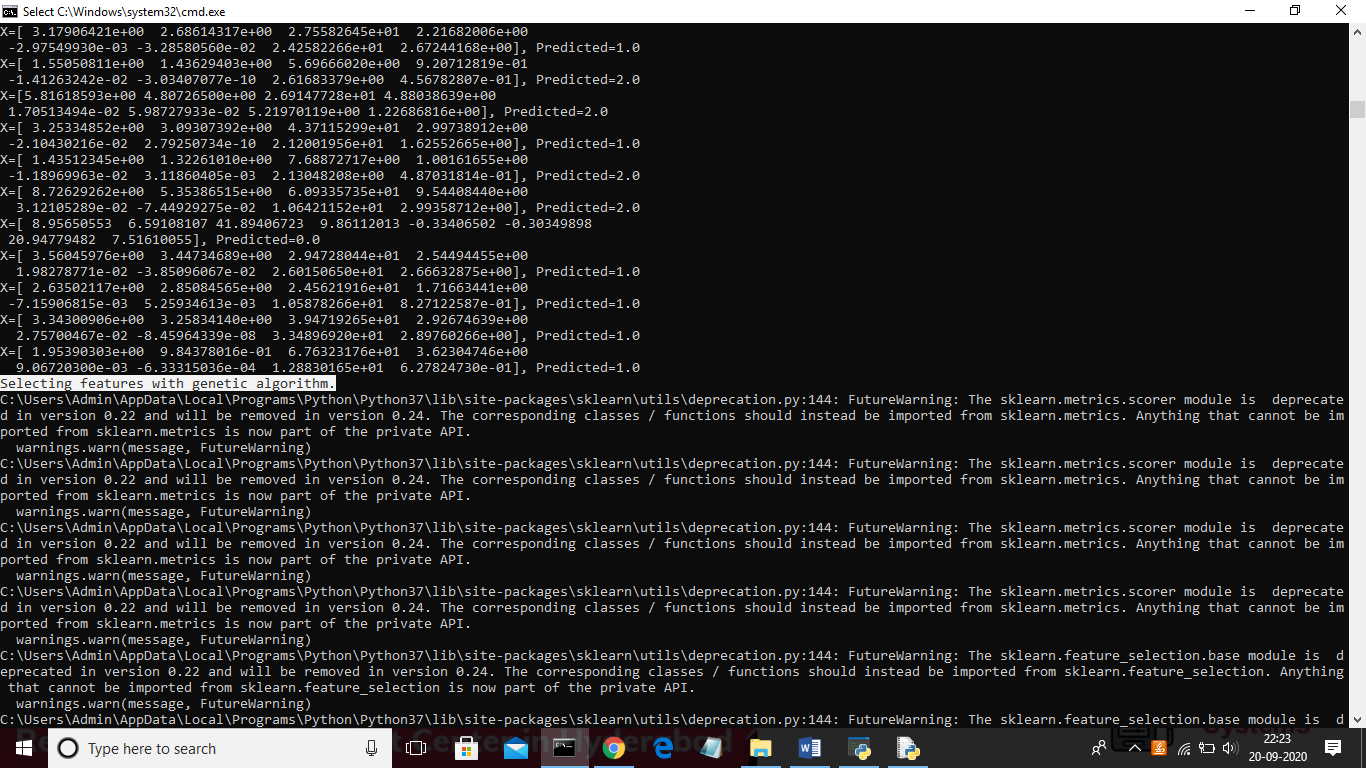


Fig.25 The calculation in command prompt

In above black console genetic algorithm starts optimal feature selection.

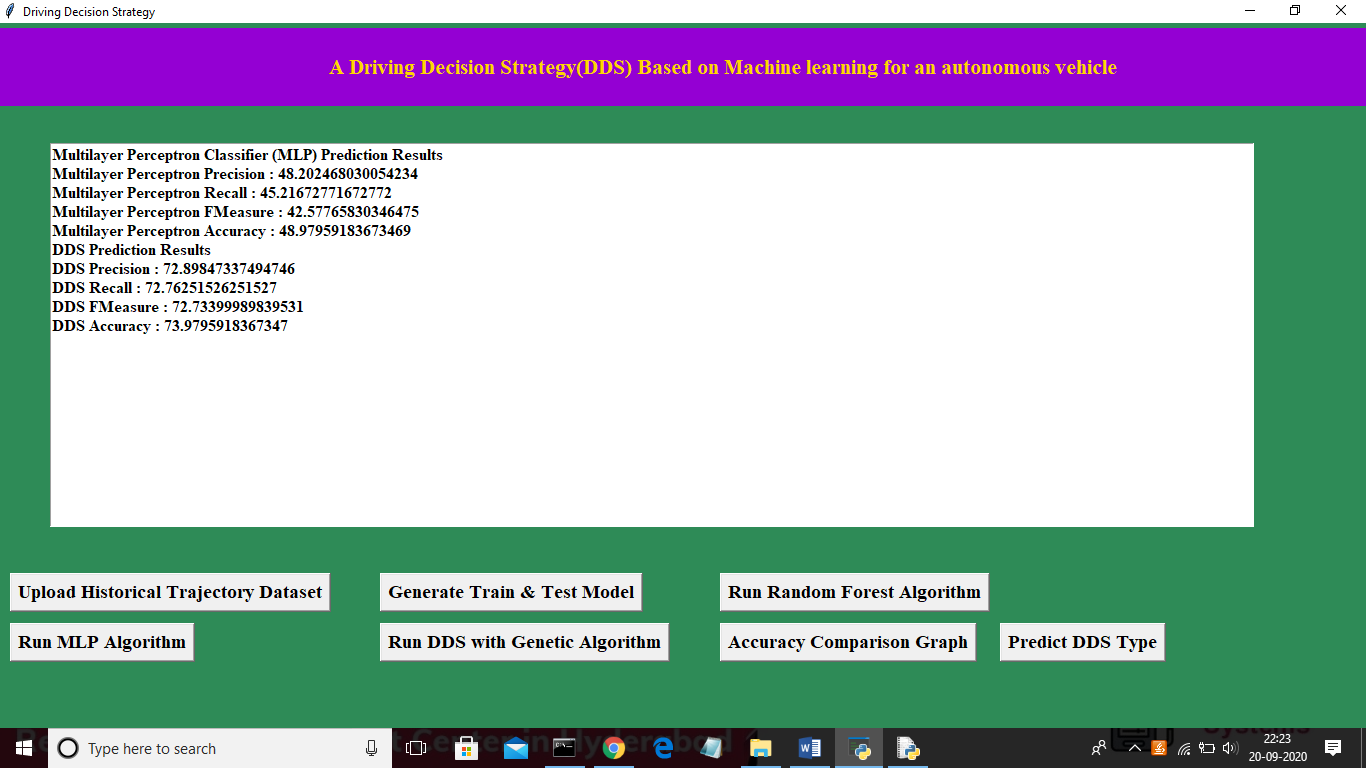


Fig.26 Running DDS with Genetic Algorithm

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

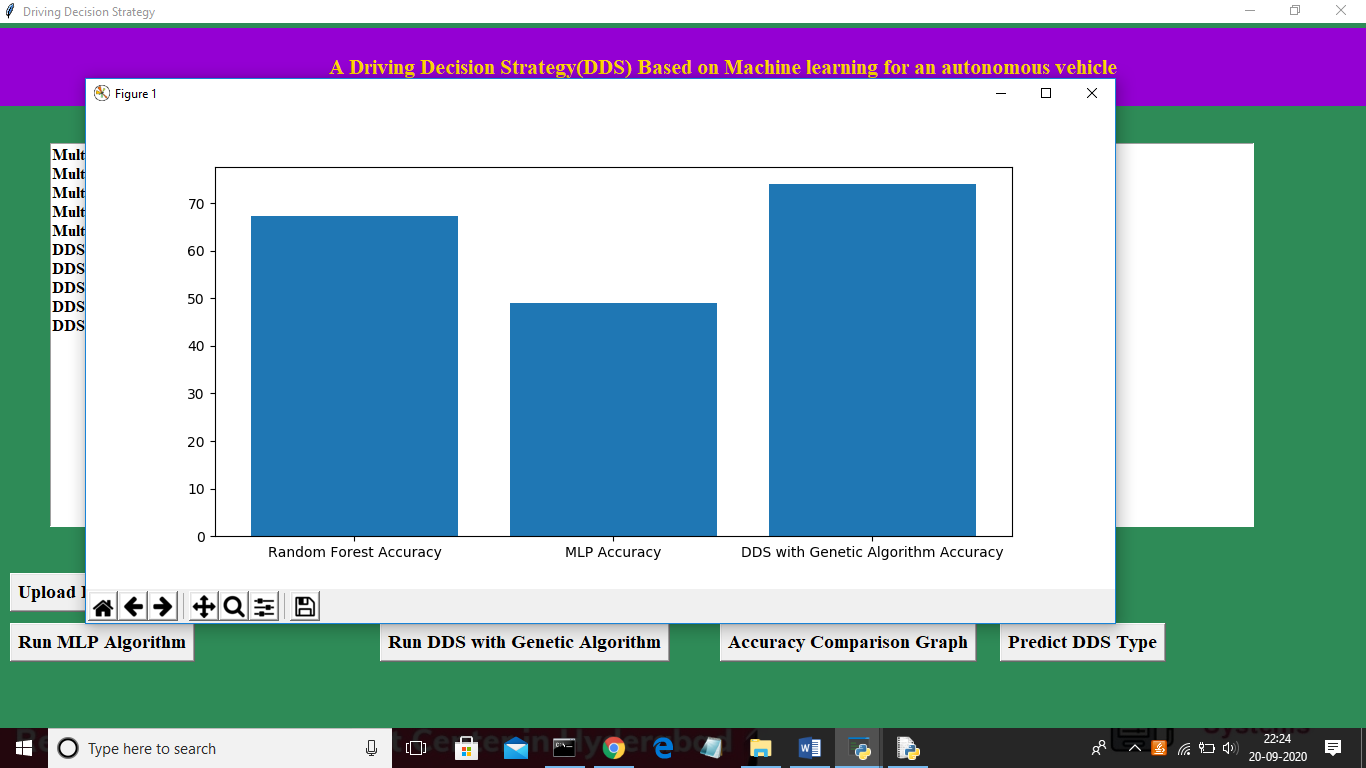


Fig.27 Accuracy Comparison Graph

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

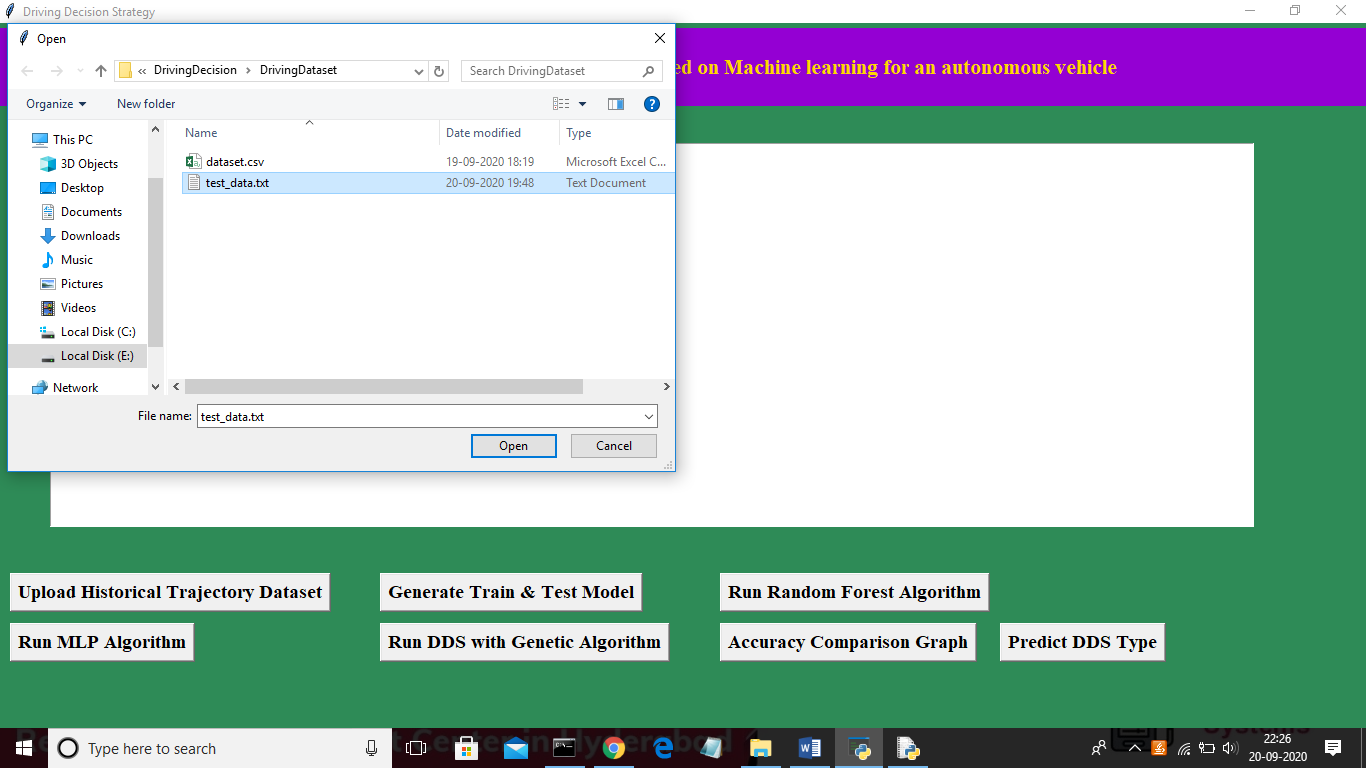


Fig.28 Predicting DDS Type

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

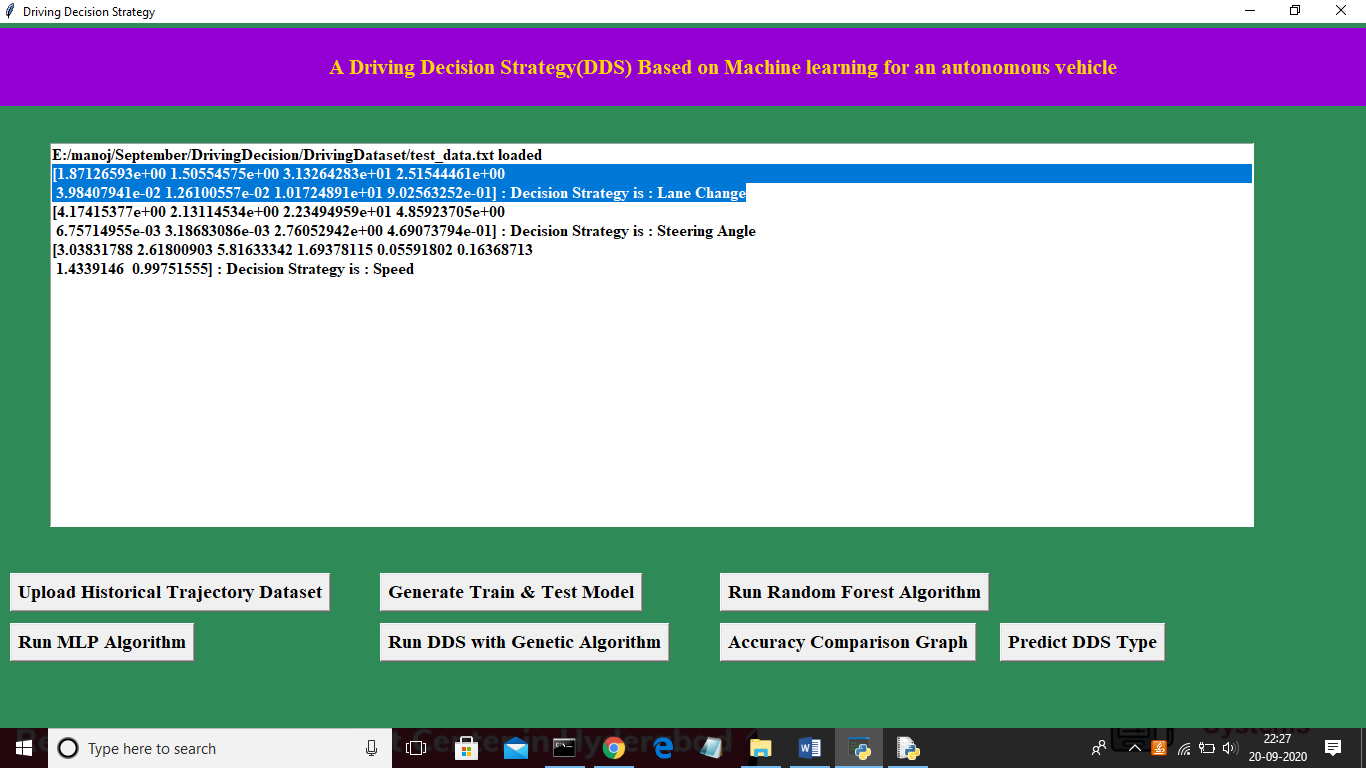


Fig.29 Predicting DDS Type

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.

**5.4 Extension Outcomes:**

In propose DDS algorithm we have used Random Forest with Genetic Algorithm and we are getting prediction accuracy up to 75% and to further enhance accuracy we are upgrading propose DDS with Genetic and XGBOOST algorithm and this extension XGBOOST algorithm giving much better accuracy compare to propose algorithm

Below is the advantage of XGBOOST algorithm

XGBoost is well known to provide better solutions than other machine learning algorithms. What makes XGBoost so popular?

1. Speed and performance: Originally written in C++, it is comparatively faster than other ensemble classifiers.
2. Core algorithm is parallelizable: Because the core XGBoost algorithm is parallelizable it can harness the power of multi-core computers. It is also parallelizable onto GPU’s and across networks of computers making it feasible to train on very large datasets as well.
3. Consistently outperforms other algorithm methods: It has shown better performance on a variety of machine learning benchmark datasets.
4. Wide variety of tuning parameters: XGBoost internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, scikit-learn compatible API etc.

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library.

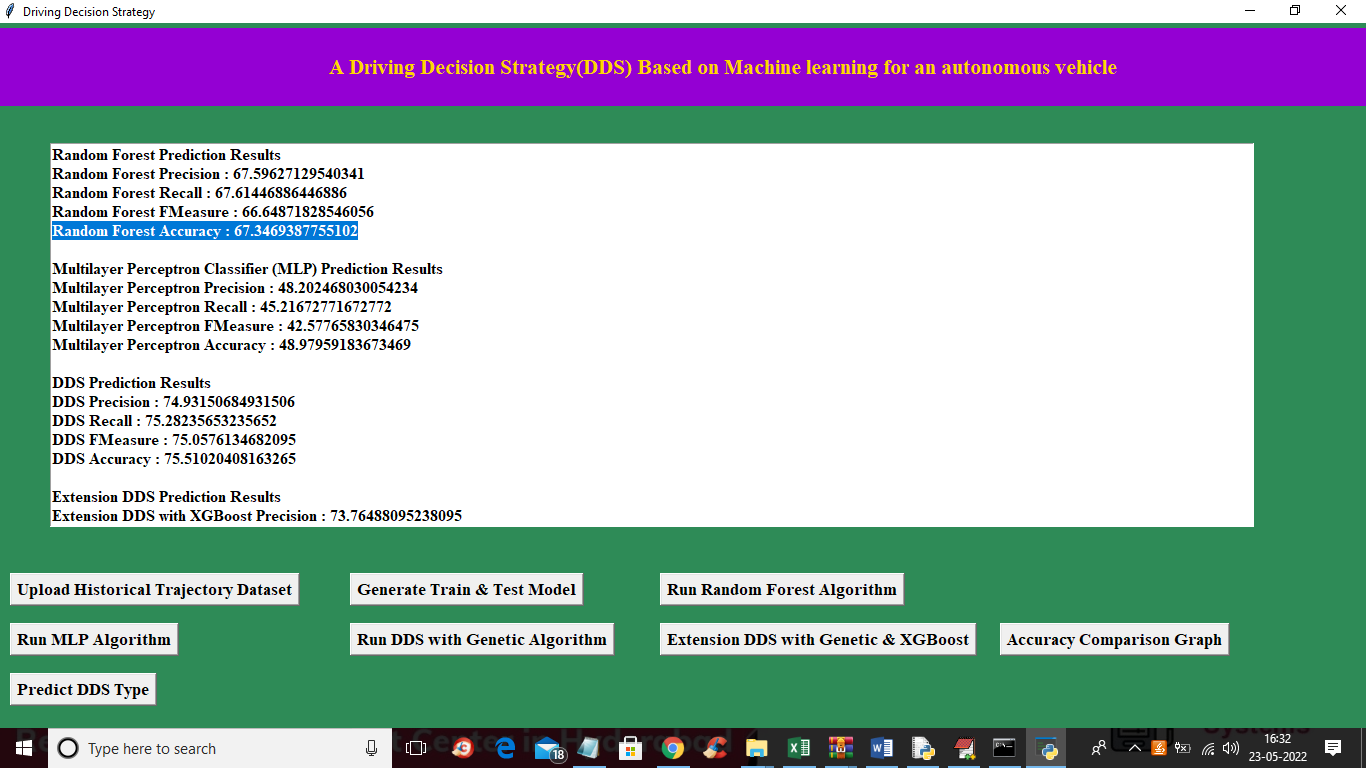


Fig.30 The screen above displays RFA,MLP, DDS and DDS with XGBoost

In above screen with Random Forest we got 67% accuracy and with Multilayer perceptron we got accuracy as 48% and with propose DDS we got accuracy as 75 and in below screen we can see accuracy of extension XGBOOST.

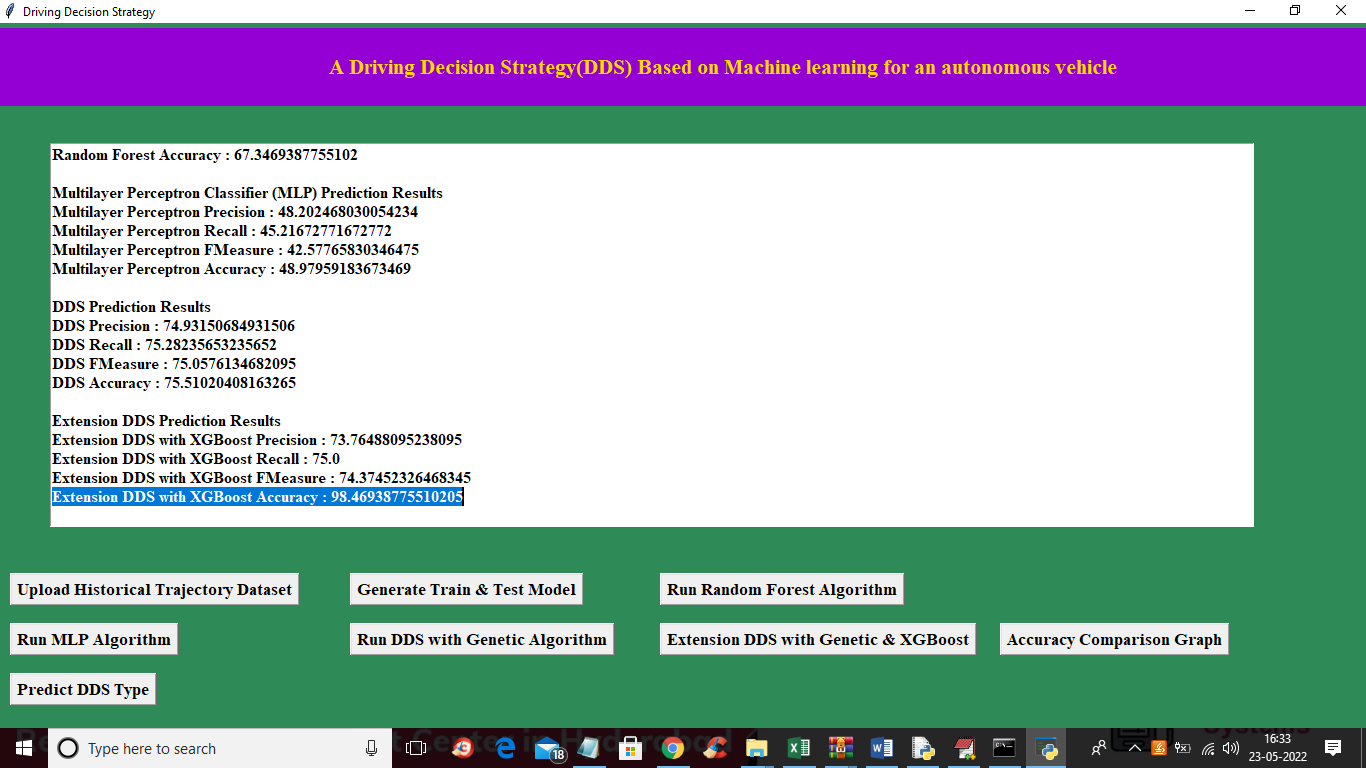


Fig.31 The screen above displays RFA,MLP, DDS and DDS with XGBoost with more data

In above screen in blue color text we can see with extension XGBOOST we got 98% accuracy and below is the comparison graph.



Fig.32 Accuracy Comparison graph with XGBoost

In above graph x-axis represents algorithm names and y-axis represents accuracy and in all algorithms extension XGBOOST has got high accuracy. Below is the detection output

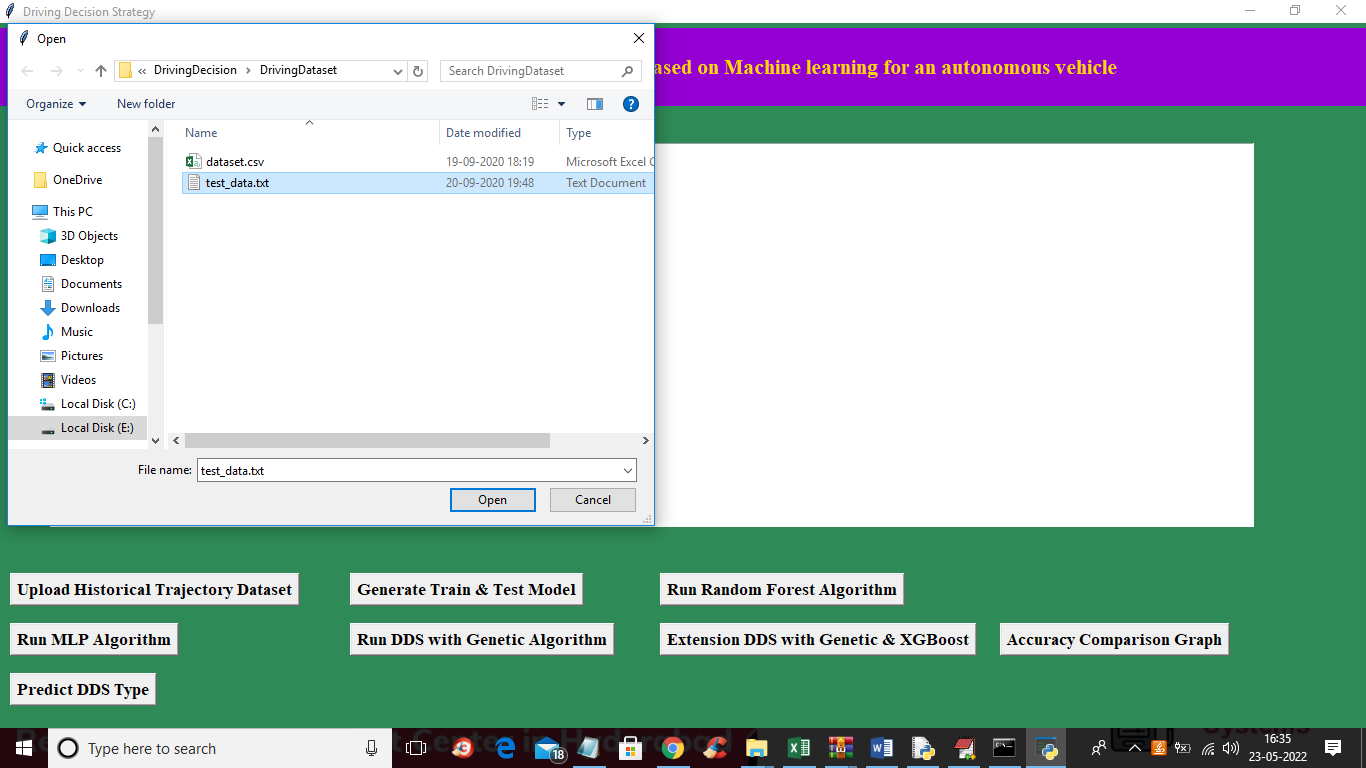
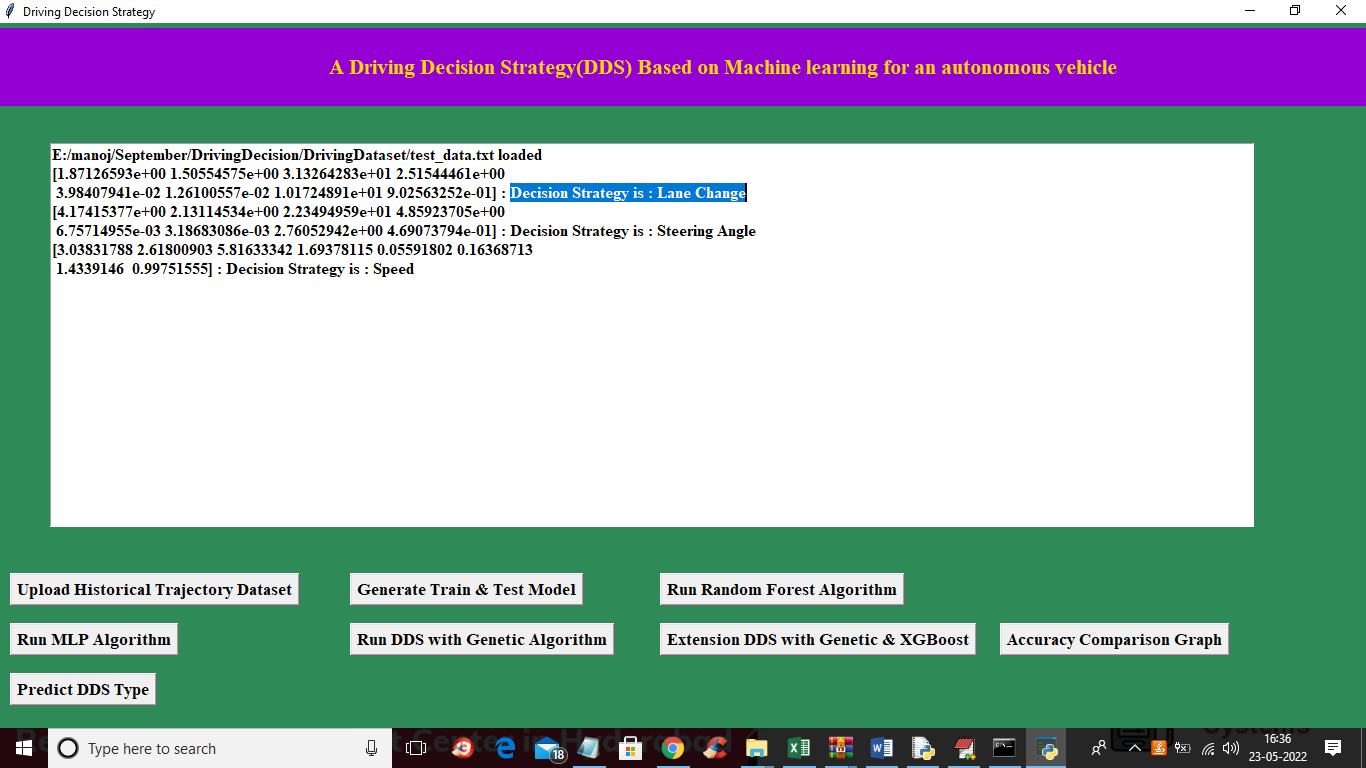


Fig.33 Uploading test data

In above screen uploading test data and below is the prediction output

 Fig.30 Prediction Of the above test data

In blue color selected text we can see prediction output as Lane Change or SPEED.

**CHAPTER 6**

**TESTING**

**6.1 About Testing:**

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

**6.2 Types of Testing:**

Types of testing are:

* Unit testing
* Integration testing
* Acceptance testing

**6.2.1 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. 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.

**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.

**6.2.2 Integration Testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

**6.2.3 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.

**Black Box Testing Techniques for Machine Learning Models:**

The following represents some of the techniques which could be used to perform black box testing on Machine Learning models:

* Model performance
* Metamorphic testing
* Dual coding
* Coverage guided fuzzing
* Comparison with simplified
* Linear models
* Testing with different data slices

1. **Model Performance**: Testing model performance is about testing the models with the test data/new data sets and comparing the model performance in terms of parameters such as accuracy/recall etc., to that of pre-determined accuracy with the model already built and moved into production. This is the most trivial of different techniques which could be used for black box testing.
2. **Metamorphic Testing**: In metamorphic testing, one or more properties are identified that represent the metamorphic relationship between input-output pairs. For example, hypothetically speaking, an ML model is built that predicts the likelihood of a person suffering from a particular disease based on different predictor variables such as age, smoking habit, gender, exercise habits, etc. Based on the detailed analysis, it is derived that given the person is a smoker and a male, the likelihood of the person suffering from the disease increases by 5% with an increase in his age by 3 years. This could be used to perform metamorphic testing as the property, age, represents the metamorphic relationship between inputs and outputs.

In metamorphic testing, the test cases that result in success lead to another set of test cases which could be used for further testing of Machine Learning models. The following represents a sample test plan:

* Given the person is a male and a smoker, determine the likelihood of the person suffering from the disease when his age is 30 years. Increase the age by 5 years. The likelihood should increase by more than 5%.
* Increase the age by 10 years. The likelihood should increase by more than 15% but less than 20%.

Test cases such as above can be executed until all results in success or failure at any step. In case, one of the test cases fail, it could result in the logging of a defect which could be dealt with, by data scientists.

1. **Dual Coding:** With dual coding technique, the idea is to build different models based on different algorithms and comparing the prediction from each of these models given a particular input data set. Let's day, a classification model is built with different algorithms such as random forest, SVM, neural network. All of them demonstrate a comparative accuracy of 90% or so with random forest showing the accuracy of 94%. This results in the selection of random forest. However, during testing, the model for quality control checks, all of the above models are preserved and input is fed into all of the models. For inputs where the majority of remaining models other than random forest gives a prediction which does not match with that of the model built with random forest, a bug/defect could be raised in the defect tracking system. These bugs could later be prioritized and dealt with by data scientists.
2. **Coverage Guided Fuzzing:** Coverage guided fuzzing is a technique where data to be fed into the Machine Learning models could be planned appropriately such that all of the features activations get tested. Take for an instance, the models built with neural networks, decision trees, random forest etc. Let's say the model is built using neural networks. The idea is to come up with data sets (test cases) which could result in the activation of each of the neurons present in the neural network. This technique sounds more like a white-box testing. However, the way it becomes part of the black box testing is the feedback which is DDS using ML Algorithms. obtained from the model which is then used to guide the further fuzzing and hence, the name — Coverage guided fuzzing. This is a work in progress.

**White Box Testing: Metrics of Machine Learning:**

* Classification accuracy
* Logarithmic Loss
* Confusion Matrix
* Area under curve
* F1 Score
* Mean Absolute error
* Mean Squared Error

1. **Classification accuracy:** Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

***A number of correct predictions***

*Accuracy = ------------------------------ ---------------—*

***Total number of predictions made***

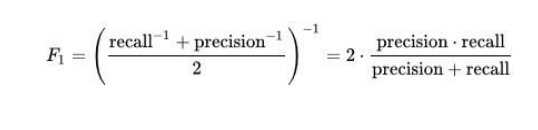
It works well only if there are equal number of samples belonging to each class. For example, consider that there are 98% samples of class A and 2% samples of class B in our training set. Then our model can easily get 98% training accuracy by simply predicting every training sample belonging to class A. When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop down to 60%.Classification Accuracy is great, but gives us the false sense of achieving high accuracy. The real problem arises, when the cost of misclassification of the minor class samples are very high. If we deal with a rare but fatal disease, the cost of failing to diagnose the disease of a sick person is much higher than the cost of sending a healthy person to more tests.

1. **Logarithmic Loss:** Log Loss is the most important classification metric based on probabilities. It's hard to interpret raw log-loss values, but log-loss is still a good metric for comparing models. For any given problem, a lower log-loss value means better predictions. Log Loss is a slight twist on something called the Likelihood Function. In fact, Log Loss is -1 \* the log of the likelihood function. So, we will start by understanding the likelihood function. This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of the true labels given a probabilistic classifier‘s predictions. The log loss is only defined for two or more labels. For a single sample with true label yt in {0,1} and estimated probability yp that,

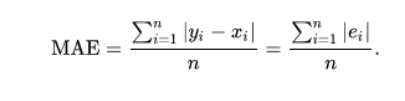
yt = 1, the log loss is -log P (yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp)) .

1. **Confusion Matrix:** Confusion matrix usage to evaluate the quality of the output of a classifier on the iris data set. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions. The figures show the confusion matrix with and without normalization by class support size (number of elements in each class). This kind of normalization can be interesting in case of class imbalance to have a more visual interpretation of which class is being misclassified. Here the results are not as good as they could be as our choice for the regularization parameter C was not the best.
2. **Area under curve:** The area under a curve between two points can be found by doing a definite integral between the two points. To find the area under the curve y = f(x) between x = a and x = b, integrate y = f(x) between the limits of a and b. Areas under the x-axis will come out negative and areas above the x-axis will be positive.

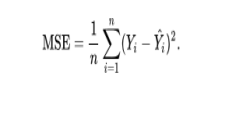
**F1 Score:** In statistical analysis of binary classification, the F1 score is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive result divided by the number of all relevant samples (all samples that should have been identified as the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.



1. **Mean Absolute Error:** In statistics, mean absolute error (MAE) is a measure of difference between two continuous variables. Assume X and Y are variables of paired observations that express the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. Consider a scatter plot of n points, where point i has coordinates (xi, yi)... Mean Absolute Error (MAE) is the average vertical distance between each point and the identity line. MAE is also the average horizontal distance between each point and the identity line. The Mean Absolute Error is given by:



1. **Mean Squared Error:** In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.



**CONCLUSION AND FUTURE ENHANCEMENTS**

In this project, we have 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.

In DDS algorithm we have used Random Forest with Genetic Algorithm and we are getting prediction accuracy up to 75% and to further enhance accuracy we are upgrading propose DDS with Genetic and XGBOOST algorithm and this extension XGBOOST algorithm giving much better accuracy compare to proposed algorithm.

Future Enhancement: Self-driving cars can help to drastically reduce car accidents as well as improve the quality of life for everyone. The concept of autonomous vehicles made the car like Tesla to have an idea of auto pilot mode. The auto pilot mode is one of the popular mechanism in Aero planes for the pilots to avoid accidents and provide a safe environment for the customers.

**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.

[5]Dataset Collected from **www.kaggle.com**