A DRIVING DECISION STRATEGY (DDS) BASED ON MACHINE LEARNING FOR AN AUTONOMOUS VEHICLE

A project report submitted to ANU, Guntur in partial fulfillment of the requirement for the award of

MASTER OF COMPUTER APPLICATIONS

Submitted By

SHAIK YASEEN
Y23MC47053

Under the esteemed guidance of

Mr.V.V.S.VENU GOPAL
Asst.Professor

DEPARTMENT OF M.C.A.

K.CHANDRAKALA P.G. COLLEGE

(Affiliated to ANU, Guntur)

Burripalem Road, Tenali, Guntur – 522 201.

(Potti Sree Ramulu Educational Society's)

K.CHANDRAKALA P.G. COLLEGE

BURRIPALEM ROAD, TENALI – 522 201.

CERTIFICATE

This is to certify that the project work entitled "A DRIVING DECISION STRATEGY (DDS) BASED ON MACHINE LEARNING FOR AN AUTONOMOUS VEHICLE" done SHAIK YASEEN is bearing Regd.No. Y23MC47053 submitted in partial fulfillment of the requirements for the award of Master of Computer Applications is a bonafide work carried out by him / her under my guidance and supervision during theacademic year 2022-24.

Internal Guide **Head of the Department**

Mr.V.V.S.VENU GOPAL **Mrs.G.BHARATHI**

External Examiner

ACKNOWLEDGEMENT

I express my sincere thanks to **Mr.G.Vijaya Krishna**, **Principal** of **K.Chandrakala P.G.College, Tenali** for giving me opportunity to do this project.

My sincere thanks to **Mrs.G.BHARATHI, Head of the Department, Dept.of M.C.A., K.Chandrakala P.G.College, Tenali,** for providing excellent environment and encouragement in completing the project.

I acknowledge with thanks the valuable guidance of **Mr.V.V.S.VENU GOPAL, Asst.Professor, Dept.of M.C.A.**, for their kind co-operation in times need.

I am very thankful to the lab faculty who always readily solved the problems during development by giving their valuable suggestions. Their encouragement assisted me in making this project a success.

I express thanks to all the teaching and non-teaching staff members of **Dept.of M.C.A., K.Chandrakala P.G.College, Tenali.**

Last but not the least I also acknowledge with humble gratitude to my parents, my team members and my dearest friends who helped me to complete this project.

SHAIK YASEEN

Y23MC47053

DECLARATION

I **SHAIK YASEEN,** declare that this project report entitled as **"A DRIVING DECISION STRATEGY (DDS) BASED ON MACHINE LEARNING FOR AN AUTONOMOUS VEHICLE"** has been prepared by me during the year **2022-24**, in partial fulfillment of the requirement, for the award of **Master of Computer Applications** in **K.Chandrakala P.G.College**, **Tenali** affiliated to **ANU, Guntur.**

I also declare that this project is the outcome of my own effort and has not been submitted toany other universities for the award of any other degree

Place: Tenali

Date:

SHAIK YASEEN
Y23MC47053

ABSTRACT

A current independent vehicle decides its driving system by thinking about just outer variables (People on foot, street conditions, and so forth.) without considering the inside state of the vehicle. To take care of the issue, this paper proposes "A Driving Decision Strategy(DDS) Based on AI for a self-governing vehicle" which decides the ideal system of a self-governing vehicle by breaking down not just the outer variables, yet additionally the inside elements of the vehicle (consumable conditions, RPM levels and so on. The DDS learns a hereditary calculation utilizing sensor information from vehicles put away in the cloud and decides the ideal driving procedure of an self-ruling vehicle. This paper contrasted the DDS and MLP what's more, RF neural system models to approve the DDS. In the analyze, the DDS had a misfortune rate around 5% lower than existing vehicle entryways and the DDS decided RPM, speed, directing point and path changes 40% quicker than the MLP also, 22% quicker than the RF...

INDEX

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 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 generatings 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.

1.2 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.

1.3 SCOPE OF THE PROJECT

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.

1.4 SOFTWARE DEVELOPMENT LIFE CYCLE

What is SDLC?

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.

SDLC is the process consisting of a series of planned activities to develop or alter the software products.

Benefits of the SDLC Process

The intent of a SDLC process it to help produce a product that is cost-efficient, effective, and of high quality. Once an application is created, the SDLC maps the proper deployment and decommissioning of the software once it becomes a legacy. The SDLC methodology usually contains the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response). Veracode makes it possible to integrate automated security testing into the SDLC process through use of its cloud based platform.

STRUCTURE OF PROJECT (SYSTEM ANALYSIS)

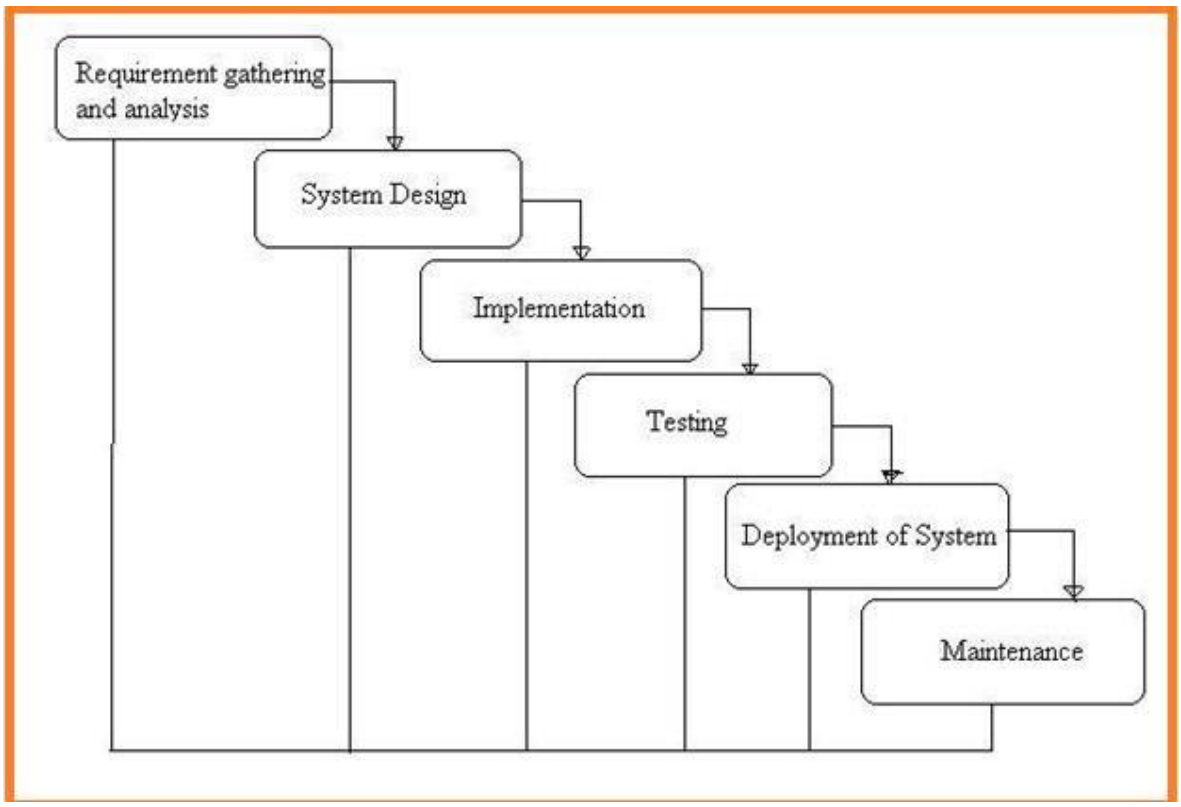


Fig: 1.4 Project SDLC

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

CHAPTER-2
LITERATURE SURVEY

2.1 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

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. : 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.

2.2 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.

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.

2.3 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

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.

2.4 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

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 model.

CHAPTER-3

SYSTEM ANALYSIS

3.1 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.

3.2 PROPOSED SYSTEM

We will propose an feature selection with MLP and RF algorithm to compute the sensor data to determine the optial driving strategy of an autonomous vehicle.

3.3 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Economical Feasibility
- Operational Feasibility
- Technical Feasibility

ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?

- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

CHAPTER-4

SYSTEM REQUIREMENTS SPECIFICATION

4.1 FUNCTIONAL REQUIREMENTS

In software engineering and systems engineering, a **functional requirement** defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.[1]

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.[2] Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[4][5]

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.[4]

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/stakeholder request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated.[6] Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use

cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

4.2 NON-FUNCTIONAL REQUIREMENT

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

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

- A non-functional requirement defines the performance attribute of a software system.
- Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
- Example of Non Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.

- Functional Requirement is a verb while Non-Functional Requirement is an attribute
- The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software

The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

CHAPTER-5

SYSTEM DESIGN

5.1 SYSTEM SPECIFICATIONS

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.

- System : Pentium i3
- Hard Disk : 500 GB.
- Monitor : 14’ Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 4 GB.

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.

- Operating system : Windows 8/10.
- Coding Language : PYTHON
- Software : Jupyter

5.2 UML DIAGRAMS

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

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

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

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

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

GOALS:

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

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

5.2.1 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.

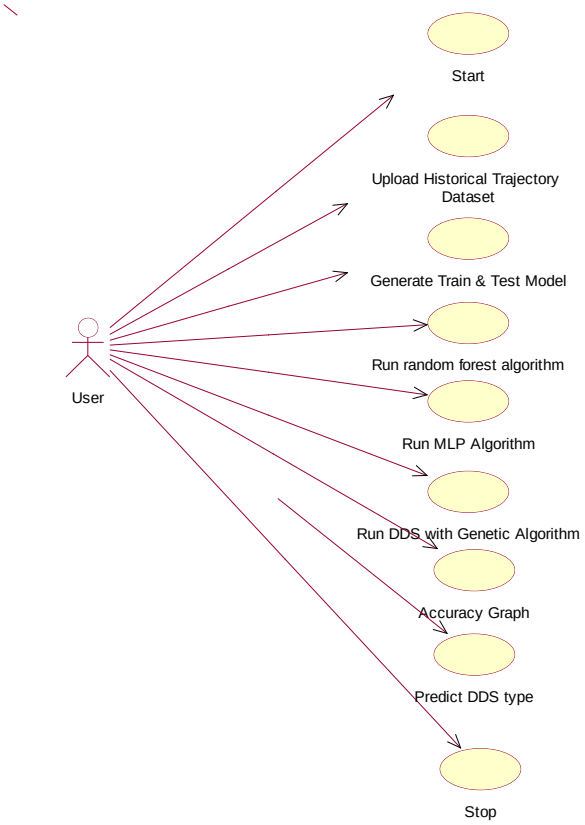


Fig 5.2.1 Use Case Diagram

5.2.2 CLASS DIAGRAM

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

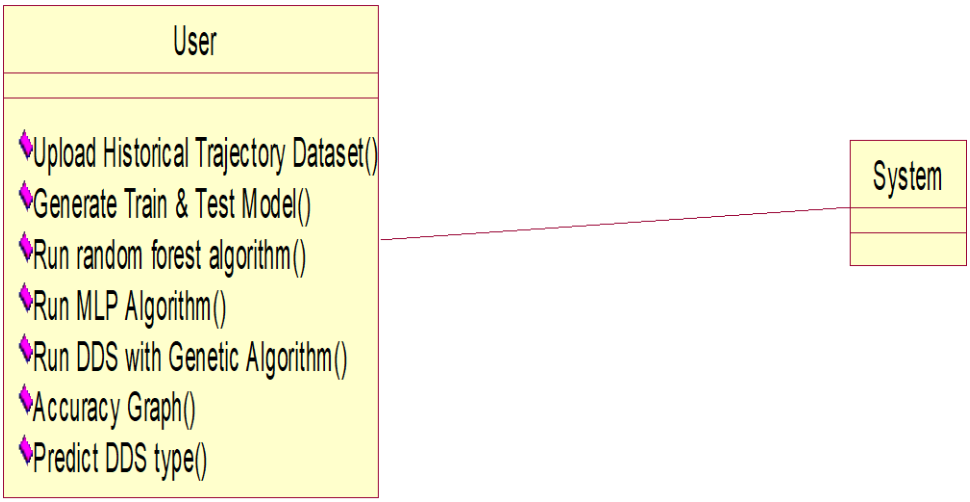


Fig 5.2.2 Class Diagram

5.2.3 SEQUENCE DIAGRAM

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

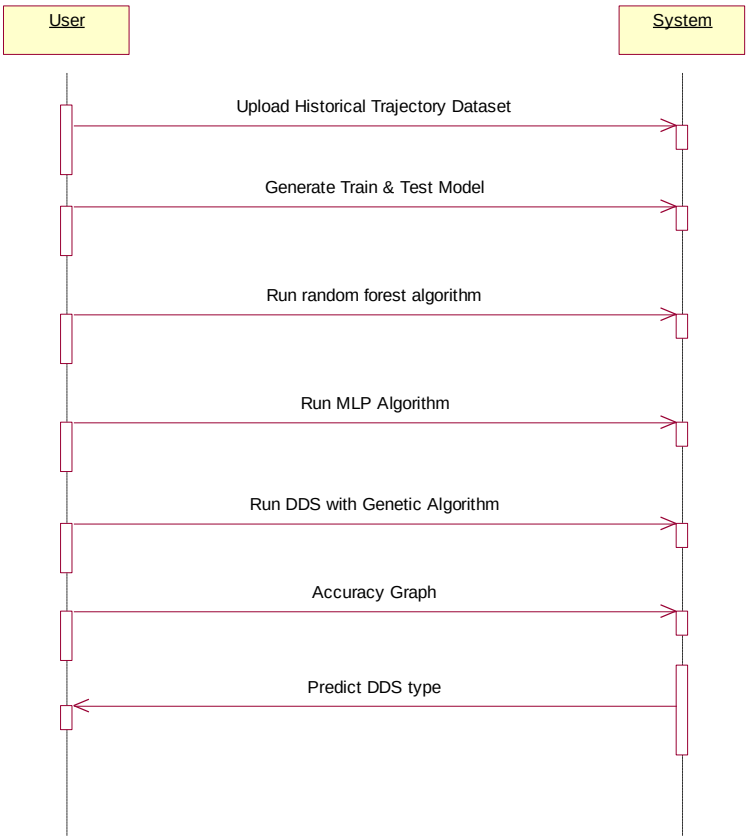


Fig 5.2.3 Sequence Diagram

5.2.4 ACTIVITY DIAGRAM:

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

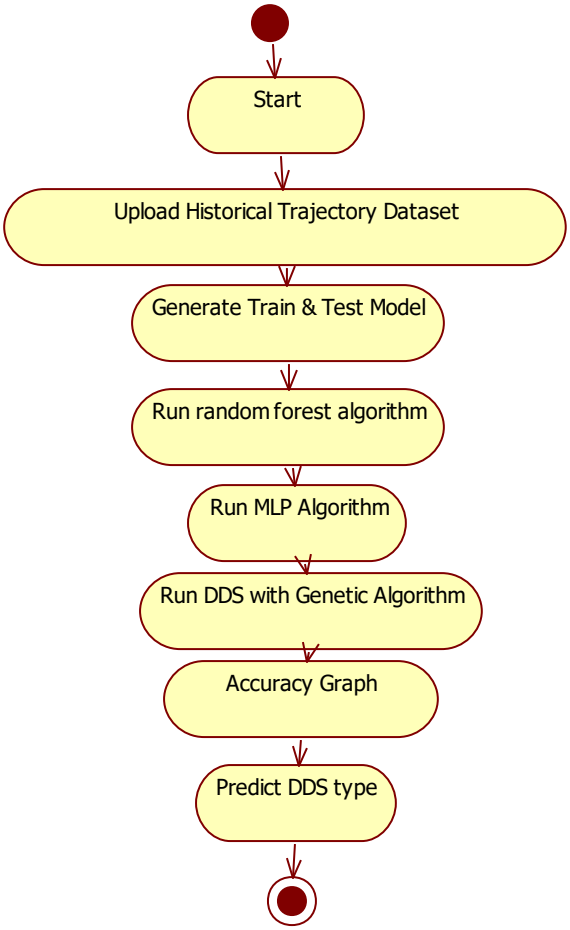


Fig 5.2.4 Activity Diagram

5.2.5 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

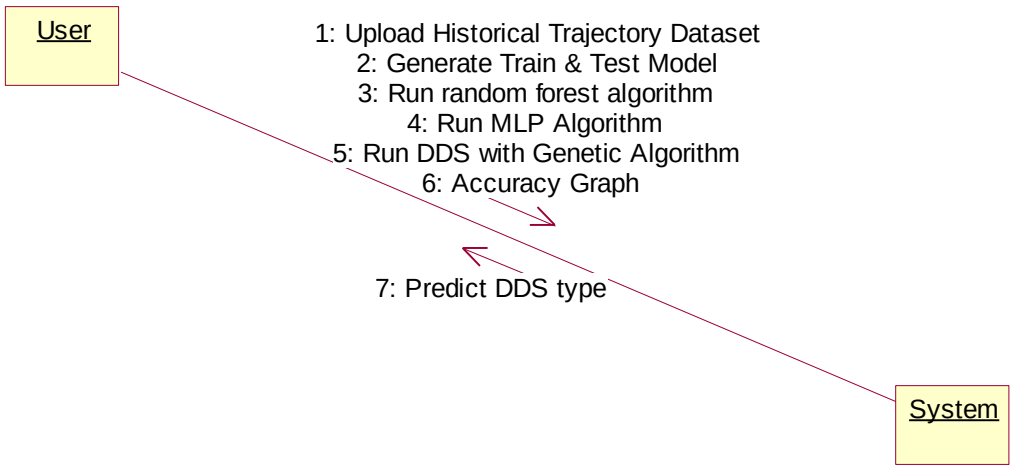


Fig 5.2.5 Collaboration Diagram

CHAPTER-6

SYSTEM IMPLEMENTATION

6.1 MODULES

- Upload Historical Trajectory Dataset
- Generate Train & Test Model
- Run Random Forest Algorithm
- Run MLP Algorithm
- Run DDS with Genetic Algorithm
- Accuracy Comparison Graph
- Predict DDS Type

6.2 SAMPLE CODE

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from genetic_selection import GeneticSelectionCV
main = tkinter.Tk()
main.title("Driving Decision Strategy") #designing main screen
main.geometry("1300x1200")
global filename
global X
le = LabelEncoder()
global mlp_acc, rf_acc, dds_acc
global classifier
def upload(): #function to driving trajectory dataset
    global filename
    filename = filedialog.askopenfilename(initialdir="DrivingDataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
def generateTrainTestData():
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
```

```
    train = pd.read_csv(filename)
    train.drop('trajectory_id', axis=1, inplace=True)
    train.drop('start_time', axis=1, inplace=True)
    train.drop('end_time', axis=1, inplace=True)
    print(train)
    train['labels'] = pd.Series(le.fit_transform(train['labels']))
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:features]
    Y = train.values[:, features]
    print(Y)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
    text.insert(END,"Dataset Length : "+str(len(X))+"\n");
    text.insert(END,"Splitted Training Length : "+str(len(X_train))+"\n");
    text.insert(END,"Splitted Test Length : "+str(len(X_test))+"\n\n");
def prediction(X_test, cls):  #prediction done here
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
# Function to calculate accuracy
```

```
def cal_accuracy(y_test, y_pred, details):
    accuracy = accuracy_score(y_test,y_pred)*100
    text.insert(END,details+"\n\n")
    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
    return accuracy
def runRandomForest():
    global rf_acc
    global classifier
    text.delete('1.0', END)
    rfc = RandomForestClassifier(n_estimators=2, random_state=0)
    rfc.fit(X_train, y_train)
    text.insert(END,"Random Forest Prediction Results\n")
    prediction_data = prediction(X_test, rfc)
    random_precision = precision_score(y_test, prediction_data,average='macro') * 100
    random_recall = recall_score(y_test, prediction_data,average='macro') * 100
    random_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    rf_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"Random Forest Precision : "+str(random_precision)+"\n")
    text.insert(END,"Random Forest Recall : "+str(random_recall)+"\n")
    text.insert(END,"Random Forest FMeasure : "+str(random_fmeasure)+"\n")
    text.insert(END,"Random Forest Accuracy : "+str(rf_acc)+"\n")
    classifier = rfc
def runMLP():
```

```
    global mlp_acc
    text.delete('1.0', END)
    cls = MLPClassifier(random_state=1, max_iter=10)
    cls.fit(X_train, y_train)
    text.insert(END,"Multilayer Perceptron Classifier (MLP) Prediction Results\n")
    prediction_data = prediction(X_test, cls)
    mlp_precision = precision_score(y_test, prediction_data,average='macro') * 100
    mlp_recall = recall_score(y_test, prediction_data,average='macro') * 100
    mlp_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    mlp_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"Multilayer Perceptron Precision : "+str(mlp_precision)+"\n")
    text.insert(END,"Multilayer Perceptron Recall : "+str(mlp_recall)+"\n")
    text.insert(END,"Multilayer Perceptron FMeasure : "+str(mlp_fmeasure)+"\n")
    text.insert(END,"Multilayer Perceptron Accuracy : "+str(mlp_acc)+"\n")
def runDDS():
    global classifier
    global dds_acc
    dds = RandomForestClassifier(n_estimators=45, random_state=42)
    selector = GeneticSelectionCV(dds,  #algorithm name
                                  cv=5,
                                  verbose=1,
                                  scoring="accuracy",
                                  max_features=5,
```

```
                    n_population=10, #population
                    crossover_proba=0.5, #cross over
                    mutation_proba=0.2,
                    n_generations=50,
                    crossover_independent_proba=0.5,
                    mutation_independent_proba=0.05, #mutation
                    tournament_size=3,
                    n_gen_no_change=5,
                    caching=True,
                    n_jobs=-1)
    selector = selector.fit(X_train, y_train)
    text.insert(END,"DDS Prediction Results\n")
    prediction_data = prediction(X_test, selector)
    dds_precision = precision_score(y_test, prediction_data,average='macro') * 100
    dds_recall = recall_score(y_test, prediction_data,average='macro') * 100
    dds_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    dds_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"DDS Precision : "+str(dds_precision)+"\n")
    text.insert(END,"DDS Recall : "+str(dds_recall)+"\n")
    text.insert(END,"DDS FMeasure : "+str(dds_fmeasure)+"\n")
    text.insert(END,"DDS Accuracy : "+str(dds_acc)+"\n")
    classifier = selector
def graph():
```

```
    height = [rf_acc, mlp_acc,dds_acc]

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

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()

def predictType():

    filename = filedialog.askopenfilename(initialdir="DrivingDataset")

    text.delete('1.0', END)

    text.insert(END,filename+" loaded\n");

    test = pd.read_csv(filename)

    test.drop('trajectory_id', axis=1, inplace=True)

    test.drop('start_time', axis=1, inplace=True)

    test.drop('end_time', axis=1, inplace=True)

    cols = test.shape[1]

    test = test.values[:, 0:cols]

    predict = classifier.predict(test)

    print(predict)

    for i in range(len(test)):

        if predict[i] == 0:

            text.insert(END,str(test[i])+" : Decision Strategy is : Lane Change\n")

        if predict[i] == 1:

            text.insert(END,str(test[i])+" : Decision Strategy is : Speed\n")
```

```
        if predict[i] == 2:
            text.insert(END,str(test[i])+" : Decision Strategy is : Steering Angle\n")
font = ('times', 16, 'bold')
title = Label(main, text='A Driving Decision Strategy(DDS) Based on Machine learning for an autonomous vehicle')
title.config(bg='darkviolet', fg='gold')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)
font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Historical Trajectory Dataset", command=upload)
uploadButton.place(x=10,y=550)
uploadButton.config(font=font1)
trainButton = Button(main, text="Generate Train & Test Model", command=generateTrainTestData)
trainButton.place(x=380,y=550)
trainButton.config(font=font1)
```

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

rfButton.place(x=720,y=550)

rfButton.config(font=font1)

mlpButton = Button(main, text="Run MLP Algorithm", command=runMLP)

mlpButton.place(x=10,y=600)

mlpButton.config(font=font1)

ddsButton = Button(main, text="Run DDS with Genetic Algorithm", command=runDDS)

ddsButton.place(x=380,y=600)

ddsButton.config(font=font1)

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

graphButton.place(x=720,y=600)

graphButton.config(font=font1)

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

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

predictButton.config(font=font1)

main.config(bg='sea green')

main.mainloop()
```

6.3 SOFTWARE 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.

6.4 TYPES OF TESTS

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. Fig.6.4 represents the black box testing:

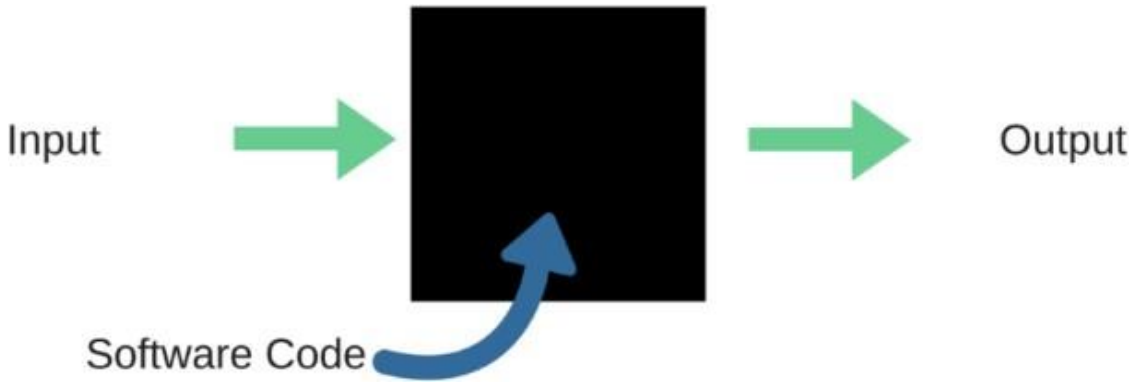


Fig 6.4 Black Box Testing

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.

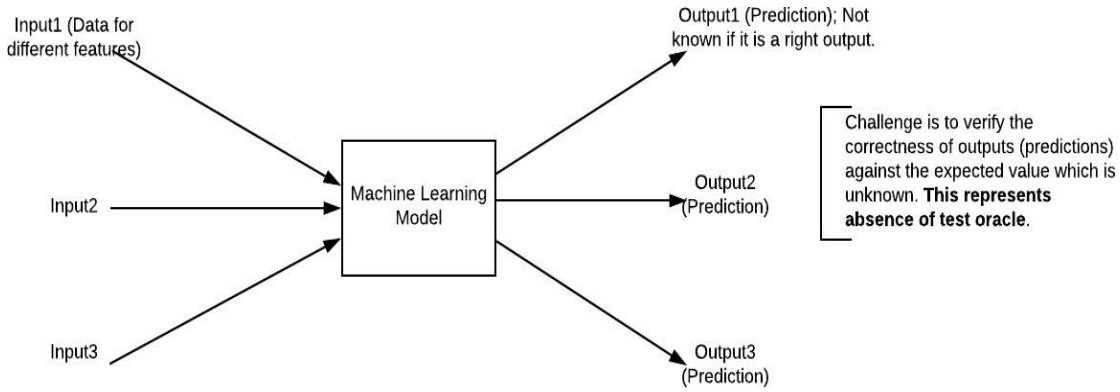


Fig. 6.5 Black Box Testing for Machine Learning algorithms

Table 6.4 Black box Testing

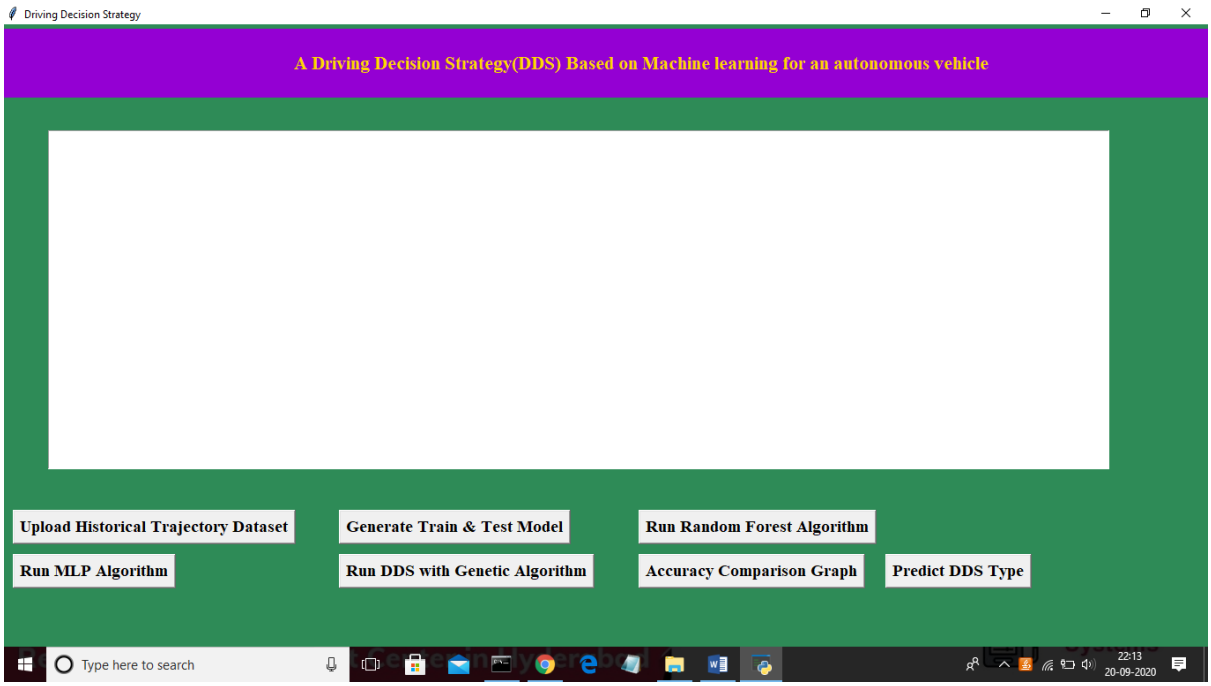
Input	**Actual Output**	**Predicted Output**
[16,6,324,0,0,0,22,0,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

The model gives out the correct output when different inputs are given which are mentioned in Table 6.4. Therefore the program is said to be executed as expected or correct program.

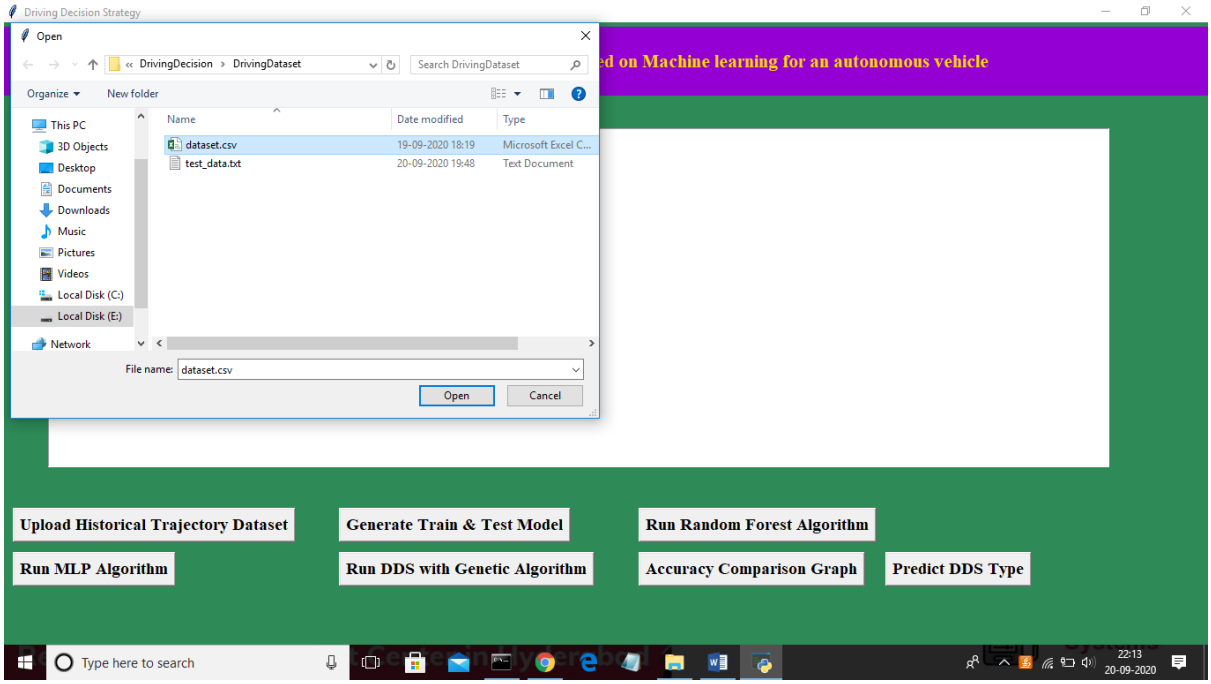
Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Start the Application	Host the Application and test if it Starts making sure the required software is Available	If it doesn't Start	We cannot run the application.	The application hosts success.	High	High
02	Home Page	Check the Deployment Environment for Properly loading the application.	If it doesn't load.	We cannot access the application.	The application is running successfully.	High	High
03	User Mode	Verify the working of The Application in freestyle Mode	If it doesn't Respond	We cannot use the Freestyle mode.	The application displays the Freestyle Page	High	High
04	Data Input	Verify if the Application takes input and updates	If it fails to take the input or store in The Database	We cannot Proceed Further	The application updates the input to application	High	High

CHAPTER-7

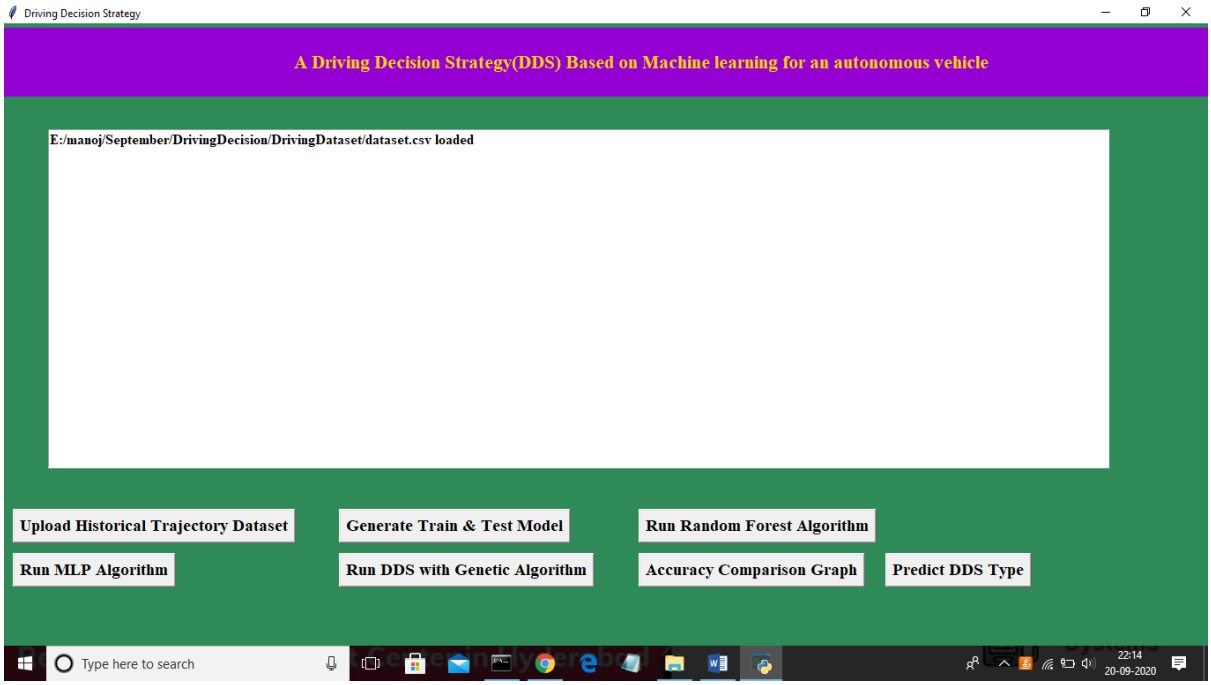
SCREEN SHOTS



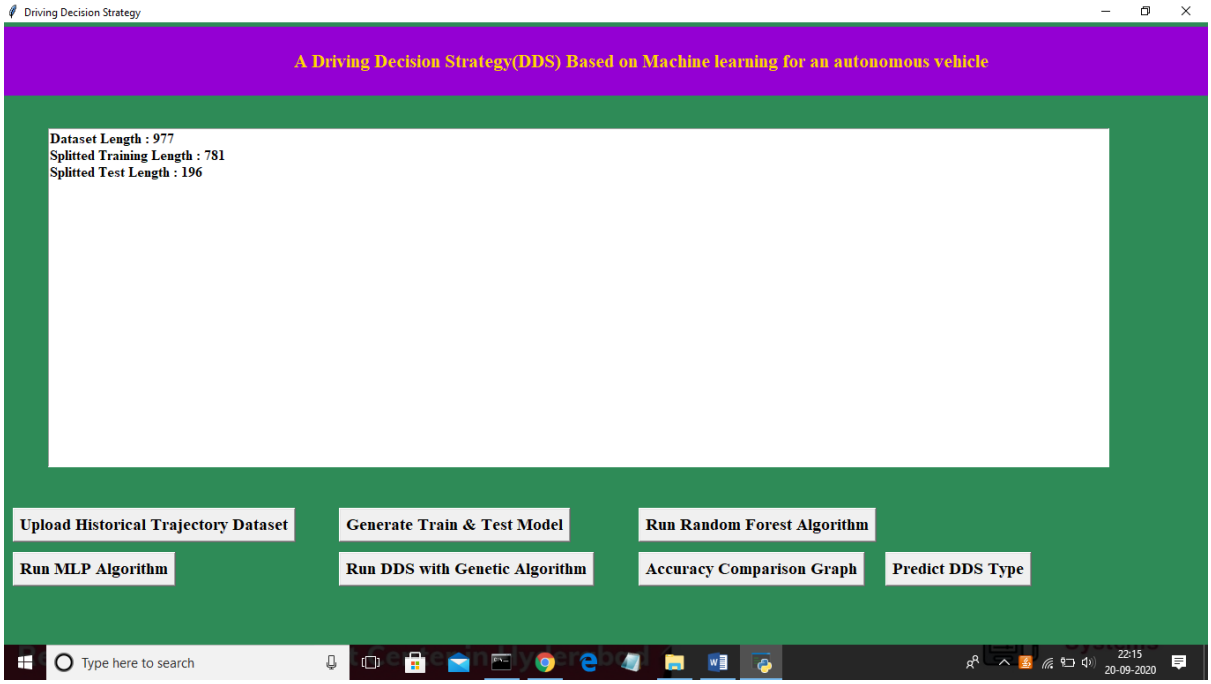
In above screen click on ‘Upload Historical Trajectory Dataset’ button and upload dataset



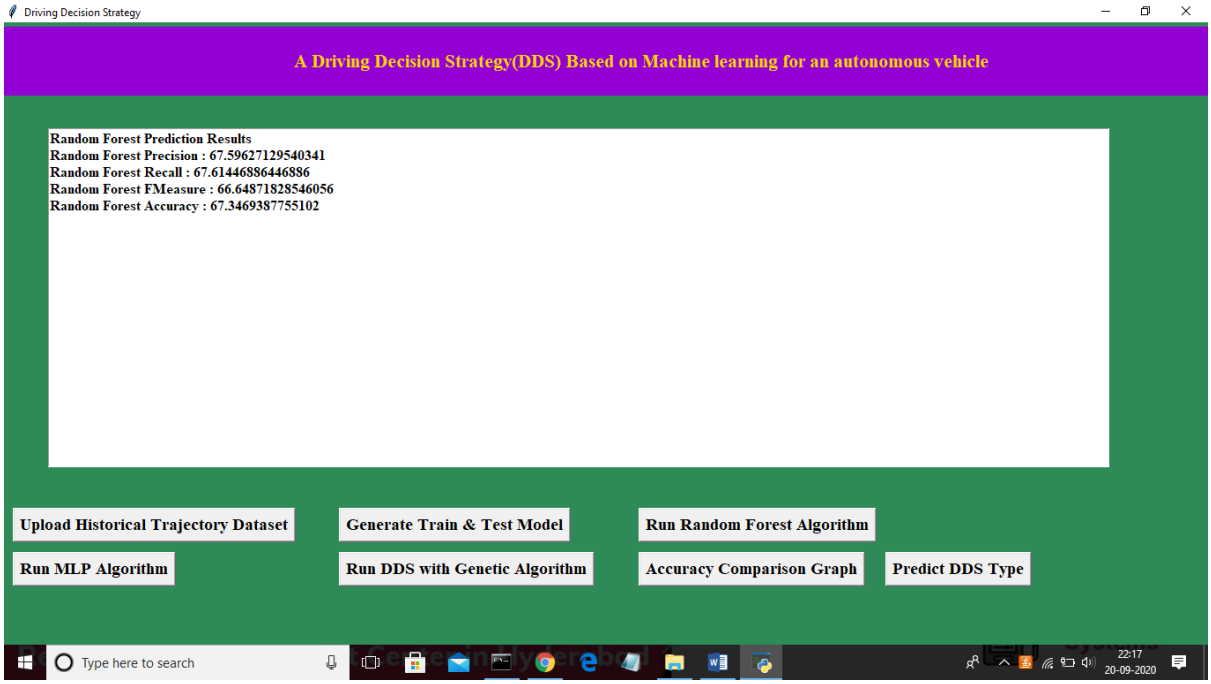
Now select ‘dataset.csv’ file and click on ‘Open’ button to load dataset and to get below screen



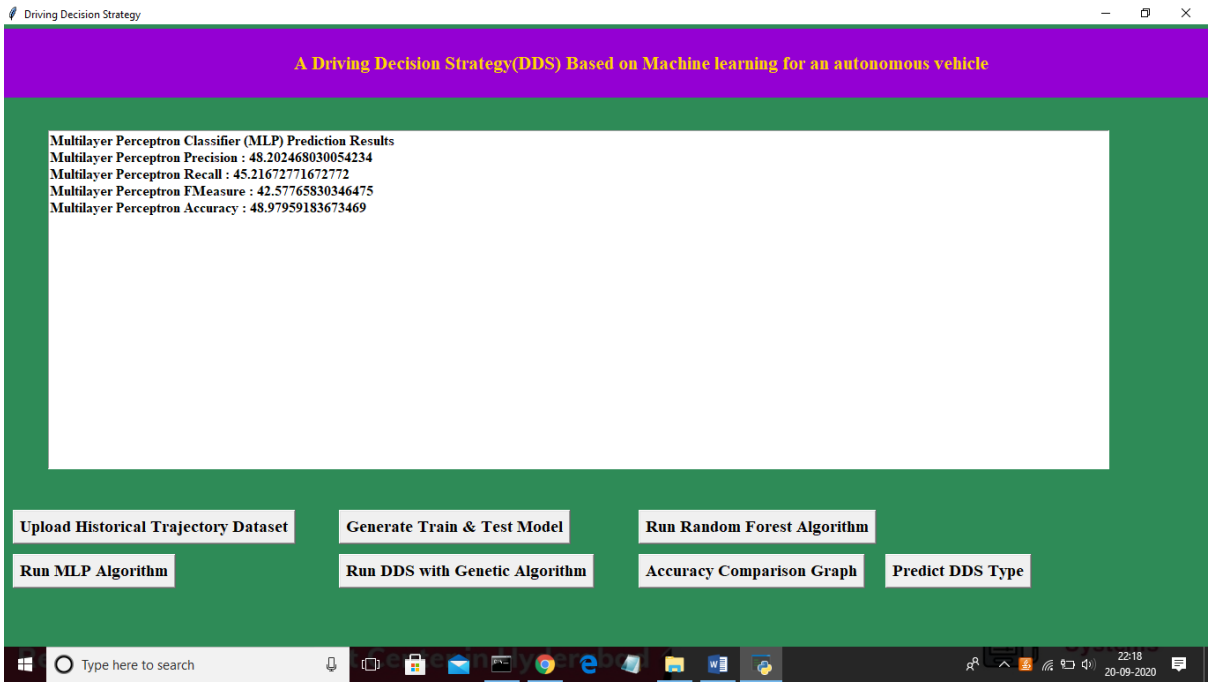
In above screen dataset is loaded and now click on ‘Generate Train & Test Model’ button to read dataset and to split dataset into train and test part to generate machine learning train model



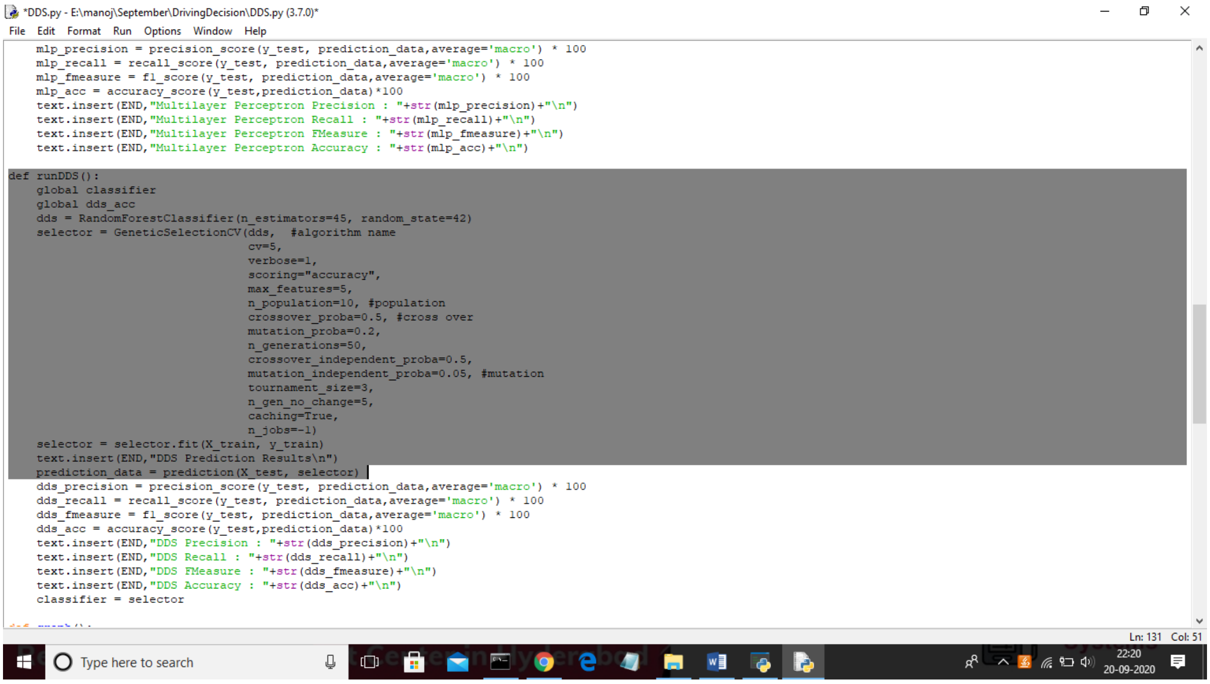
In above screen dataset contains 977 total trajectory records and application using 781 (80% of dataset) records for training and 196 (20% of dataset) for testing. Now both training and testing data is ready and now click on 'Run Random Forest Algorithm' button to train random forest classifier and to calculate its prediction accuracy on 20% test data.



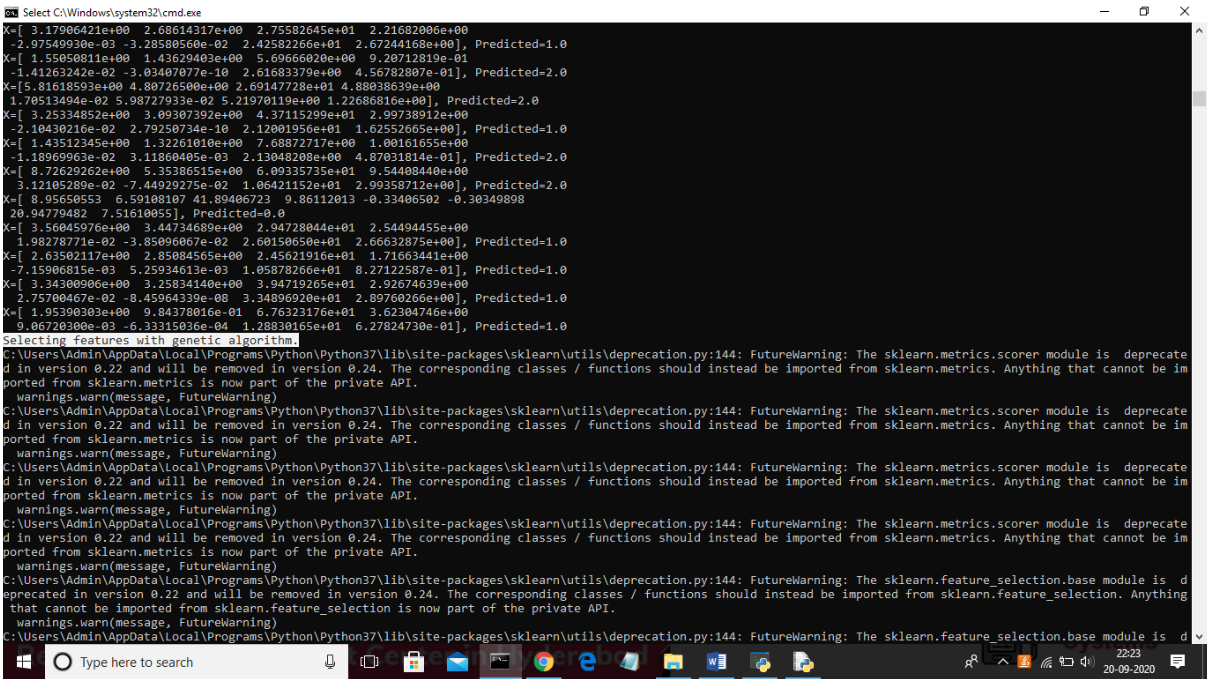
In above screen we calculated random forest accuracy, precision, recall and fmeasure and random forest got 67% prediction accuracy. Now click on 'Run MLP Algorithm' button to train MLP model and to calculate its accuracy



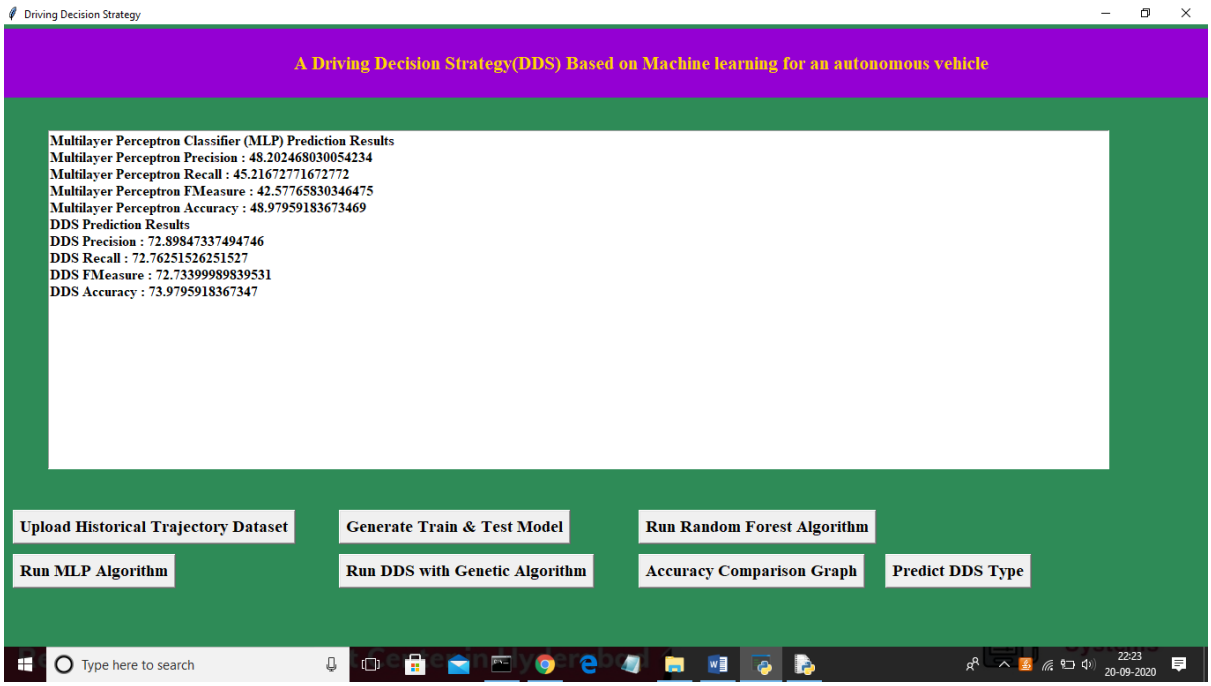
In above screen MLP got 48% prediction accuracy and in below screen we can see genetic algorithm code used for building propose DDS algorithm



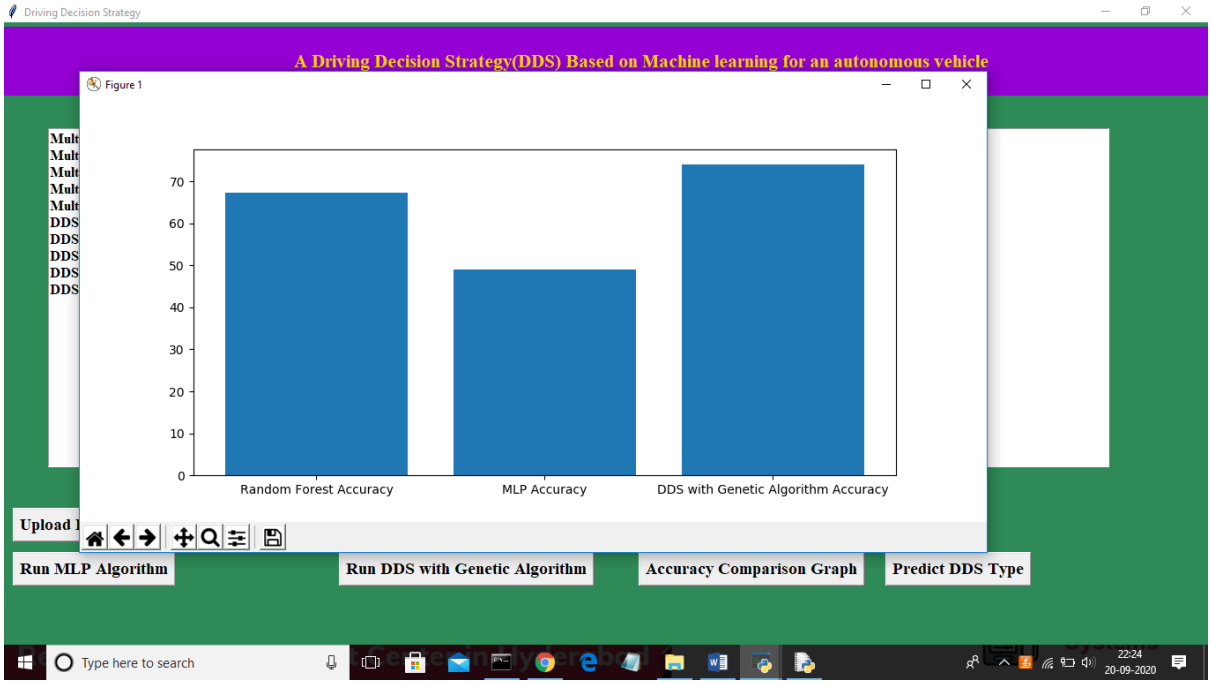
In above screen we can see genetic algorithm code used in DDS algorithm and now click on ‘Run DDS with Genetic Algorithm’ button to train DDS and to calculate its prediction accuracy



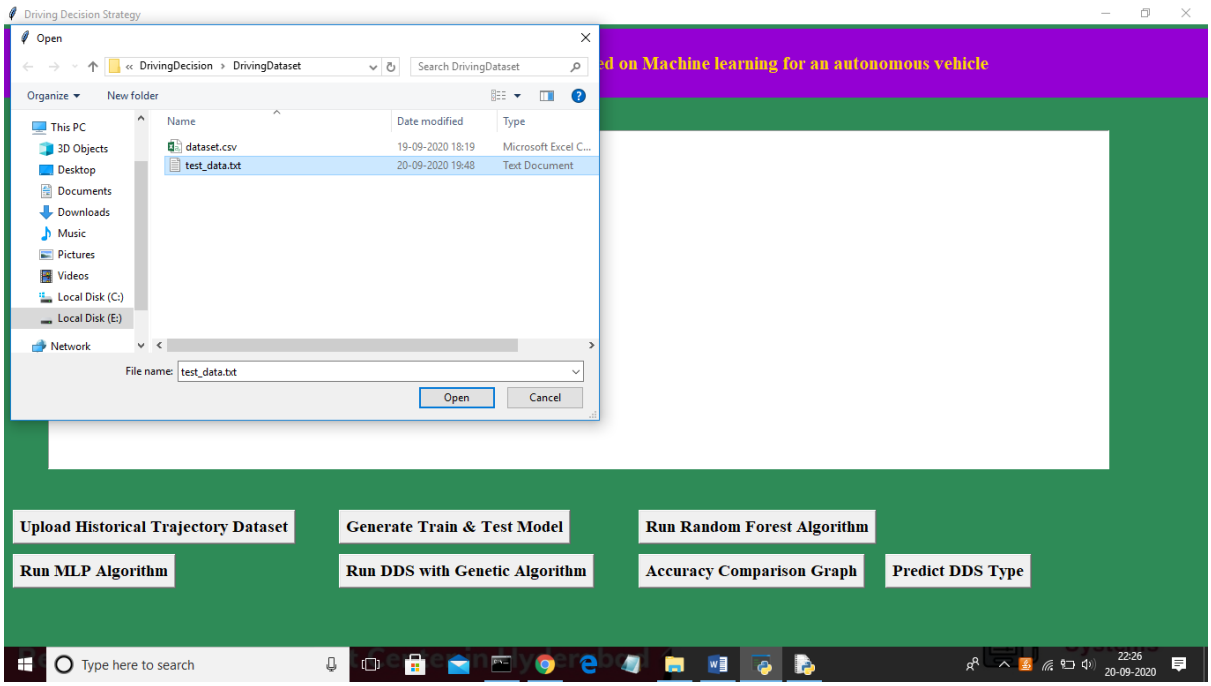
In above black console genetic algorithm starts optimal feature selection



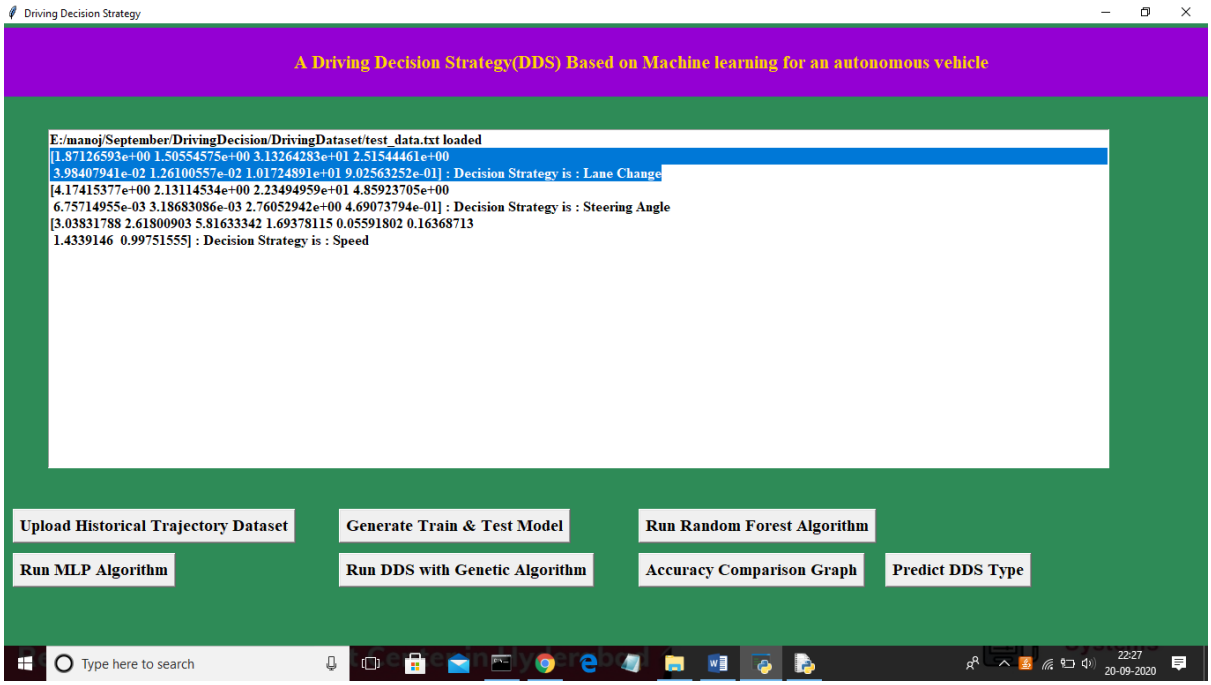
In above screen propose DDS algorithm got 73% prediction accuracy and now click on 'Accuracy Comparison Graph' button to get below graph



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.

CHAPTER-8

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.

APPENDIX-A

BIBLIOGRAPHY

[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

TECHNOLOGY USED

Python Introduction

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

Python History

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.

- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
    - ABC language.
    - Modula-3

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at offical web address. The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifing applications areas where python can be applied.

1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

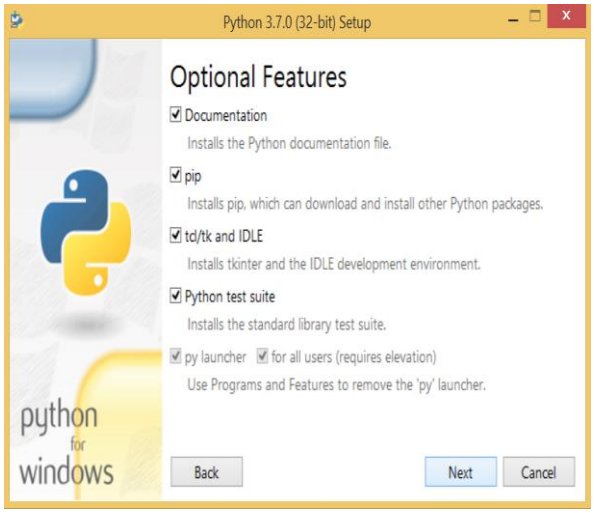
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Installation on Windows

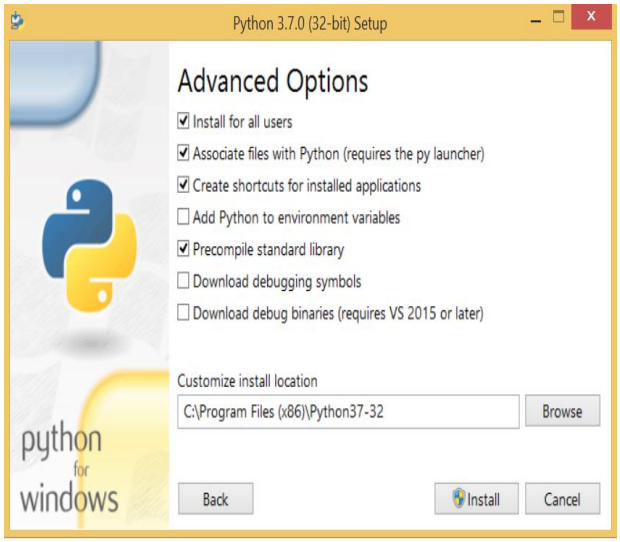
Visit the link *https://www.python.org/downloads/* to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.

Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.

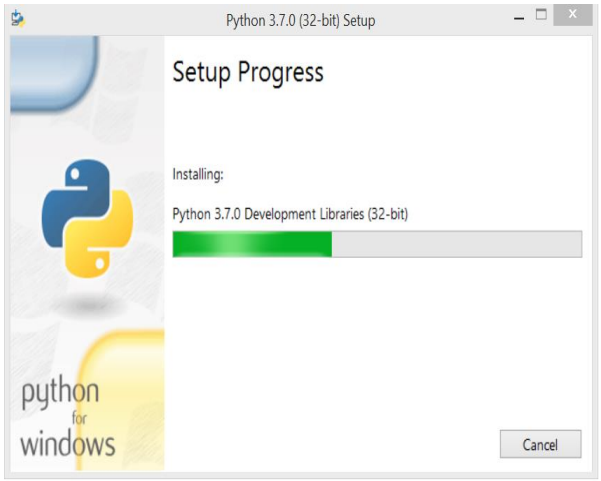
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



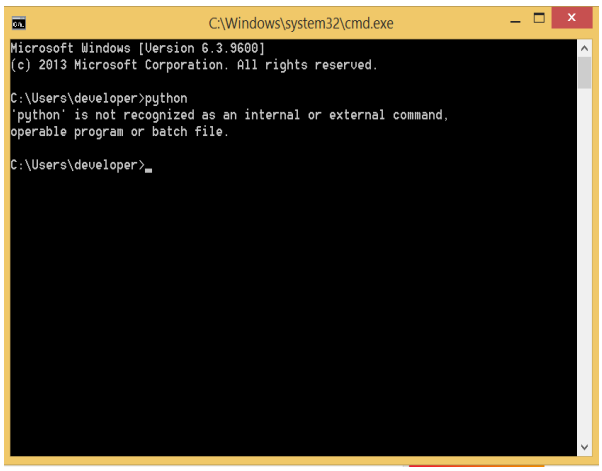
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



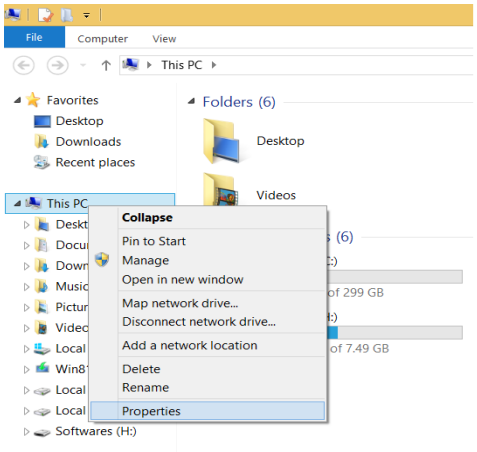
Now, we are ready to install python-3.6.6. Lets install it.

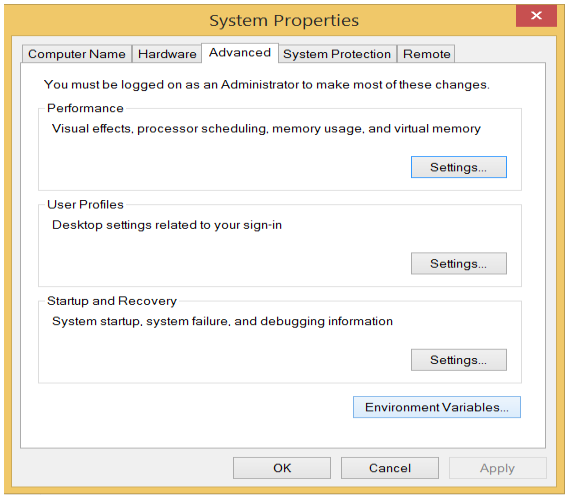


Now, try to run python on the command prompt. Type the command python in case of python2 or python3 in case of python3. It will show an error as given in the below image. It is because we haven't set the path.

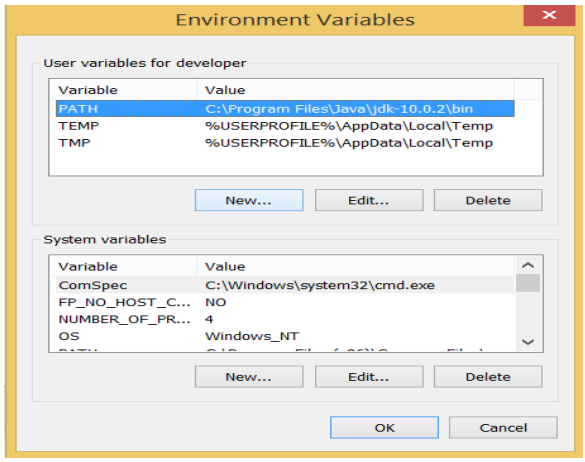


To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

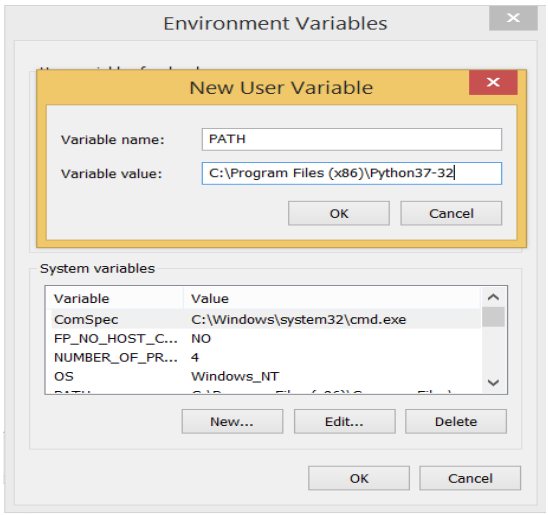




Add the new path variable in the user variable section.



Type PATH as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type python again. It will open the python interpreter shell where we can execute the python statements.

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

```
>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>
```

Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
(tutorial-env) $ pip search astronomy
skyfield            - Elegant astronomy for Python
gary                - Galactic astronomy and gravitational dynamics.
novas               - The United States Naval Observatory NOVAS astronomy library
astroobs            - Provides astronomy ephemeris to plan telescope observations
PyAstronomy         - A collection of astronomy related tools for Python.
...
```

pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package's name:

```
(tutorial-env) $ pip install novas
Collecting novas
Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: novas
Running setup.py install for novas
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(tutorial-env) $ pip install requests==2.6.0
```

Collecting requests==2.6.0
Using cached requests-2.6.0-py2.py3-none-any.whl
Installing collected packages: requests
Successfully installed requests-2.6.0
If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:
(tutorial-env) $ pip install --upgrade requests
Collecting requests
Installing collected packages: requests
Found existing installation: requests 2.6.0
Uninstalling requests-2.6.0:
Successfully uninstalled requests-2.6.0
Successfully installed requests-2.7.0
pip uninstall followed by one or more package names will remove the packages from the virtual environment.
pip show will display information about a particular package:
(tutorial-env) $ pip show requests

Metadata-Version: 2.0
Name: requests
Version: 2.7.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.com
License: Apache 2.0
Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages
Requires:
pip list will display all of the packages installed in the virtual environment:
(tutorial-env) $ pip list
novas (3.1.1.3)
numpy (1.9.2)
pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.