A

Mini Project

On

A Driving Decision Strategy(DDS Based on Machine Learning for an Autonomous Vehicle

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

B.Venkata Bhargav (217R1A0509)

T.Madhu (217R1A0556)

M.Abhishek(217R1A0536)

Under the Guidance of

Mr.K.Praveen Kumar

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**A Driving Decision Strategy(DDS) based on Machine Learning for Autonomous Vehicle**" being submitted by **B.VENKATA BHARGAV (217R1A0509), T.MADHU(217R1A0556) & M.ABHISHEK(217R1A0536)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2024-25.

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

Mr.K.Praveen Kumar
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. N.Bhaskar
HoD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide

Mr. K. Praveen Kumar, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Dr. J. Narasimharao, Dr. K. Maheswari, Dr. K. Suma** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. N. Bhaskar,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We would like to express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

B.VENKATA BHARGAV (217R1A0509)
T.MADHU (217R1A0556)
M.ABHISHEK (217R1A0536)

ABSTRACT

A modern-day self-sustaining car determines its driving method by means of thinking about solely exterior factors(Pedestrians, street conditions, etc.) barring thinking about the interior circumstance of the vehicle. To overcome above problems, in this paper author proposed a new strategy i.e A Driving Decision Strategy(DDS) Based on Machine learning for an autonomous vehicle" Analysis of both external and internal factors determines the optimal strategy for an autonomous vehicle (consumable conditions, RPM levels etc.). To implement this, the project author has introduced and 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 passes to genetic algorithm to choose optimal value which helps in faster and efficient prediction. Propose DDS with genetic algorithm performance is comparing with existing machine learning algorithm such as Random Forest and MLP (multilayer perceptron algorithm.). Propose DDS shows better prediction accuracy compare to random forest and MLP.

LIST OF FIGURES

i

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

Autonomous vehicles are becoming more popular, and technology is advancing rapidly. Machine learning is a critical component of the decision-making process for autonomous vehicles. Machine learning algorithms can help autonomous vehicles to leam from real-world data and improve their decision-making capabilities. They can analyze data from various sensors, including cameras, LIDAR, and radar, to detect and recognize objects, such as other vehicles, pedestrians, and road signs. Machine learning can also help autonomous vehicles to predict potential hazards and plan their routes accordingly. In addition, machine learning can help to optimize the energy consumption of autonomous vehicles, reduce traffic congestion, and improve overall safety on the roads.

1.2 PROJECT PURPOSE

The purpose of this project is to advance the capabilities of autonomous vehicles by integrating a machine learning-based decision-making framework that can enhance real-time driving strategies. With the increasing need for automation in transportation, this system aims to reduce human intervention by using predictive algorithms to decide vehicle movements like lane changes, steering adjustments, and speed control. The project's core goal is to ensure that autonomous vehicles can operate efficiently and safely by analyzing vehicle data and making intelligent, data-driven decisions.

1.3 PROJECT FEATURES

This project offers a range of features aimed at improving the decision-making process for autonomous vehicles. At the core of the system is the ability to collect and process vehicle data such as speed, RPM, and steering angles in real time, providing the necessary inputs for accurate predictions. The system incorporates advanced machine learning models, including Random Forests, Neural Networks, and Genetic Algorithms, to train on historical driving data and predict driving decisions in real time. One of the main features is its predictive capability, where it analyzes the current driving conditions and forecasts actions such as lane changes and speed adjustments.

2. SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

With the advancement in technology, the development of autonomous vehicles has become an essential area of research. Autonomous vehicles are vehicles that are capable of sensing their environment and navigating without human input. One of the crucial components of an autonomous vehicle is its ability to make driving decisions in real time. The driving decision strategy must be capable of handling various scenarios and provide safe and efficient driving. Therefore, the problem statement is to develop a driving decision strategy using machine learning techniques that can provide safe and efficient driving in various driving scenarios for an autonomous vehicle. The solution should be able to handle uncertain and dynamic environments, ensure passengers’ safety, and meet the regulatory requirements for an autonomous.

2.2 EXISTING SYSTEM

In current autonomous vehicle systems, decision-making is typically handled by rule-based algorithms that depend on sensor data to control driving actions. These systems, while effective in controlled environments, struggle with more complex, real-world driving scenarios. The existing approach often lacks the flexibility and predictive power to account for rapid changes in the driving environment, such as sudden lane changes or unexpected obstacles. Furthermore, these systems are limited in their ability to compare multiple decision-making models, which restricts the opportunity for optimization and improvement in driving performance.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Reactive rather than predictive decision-making.
- Limited flexibility in handling dynamic driving conditions.
- Inability to compare and evaluate multiple decision-making models.
- Dependence on pre-defined rules, which may not adapt well to unforeseen scenarios.
- Time consuming.
- Needs manual calculations.

2.3 PROPOSED SYSTEM

The proposed **Driving Decision Strategy (DDS)** addresses the limitations of current systems by incorporating machine learning models to improve decision-making capabilities. The system will analyze vehicle data (such as speed, RPM, steering angle) and predict optimal driving decisions based on historical data and real-time inputs. Machine learning models will be trained to recognize driving patterns and adapt to changing conditions, making the system proactive rather than reactive. The DDS will also allow for the comparison of various models (such as Random Forest, MLP Algorithm, etc.) to select the best performing one, further improving decision accuracy.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- **Predictive Decision-Making**: The system will anticipate driving manoeuvre's based on data patterns, making decisions faster and more reliable.
- **Adaptability**: The machine learning models can adapt to changing driving conditions, ensuring better performance in complex scenarios.
- **Model Comparison**: The system will provide performance metrics and visual comparisons between different machine learning models, ensuring continuous optimization.
- **Improved Safety**: By making more accurate and faster decisions, the DDS can improve overall vehicle safety.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIOURAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- **Processor**: Intel i5 or above.
- **Memory**: 8GB RAM or higher.
- **Storage**: 16GB or above.

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements:

- Operating system : Windows 8 and Above
- Languages : Python 3.7.0
- IDE : VS Code

3. ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for breed detectionusing machine learning, starting from input to final prediction.

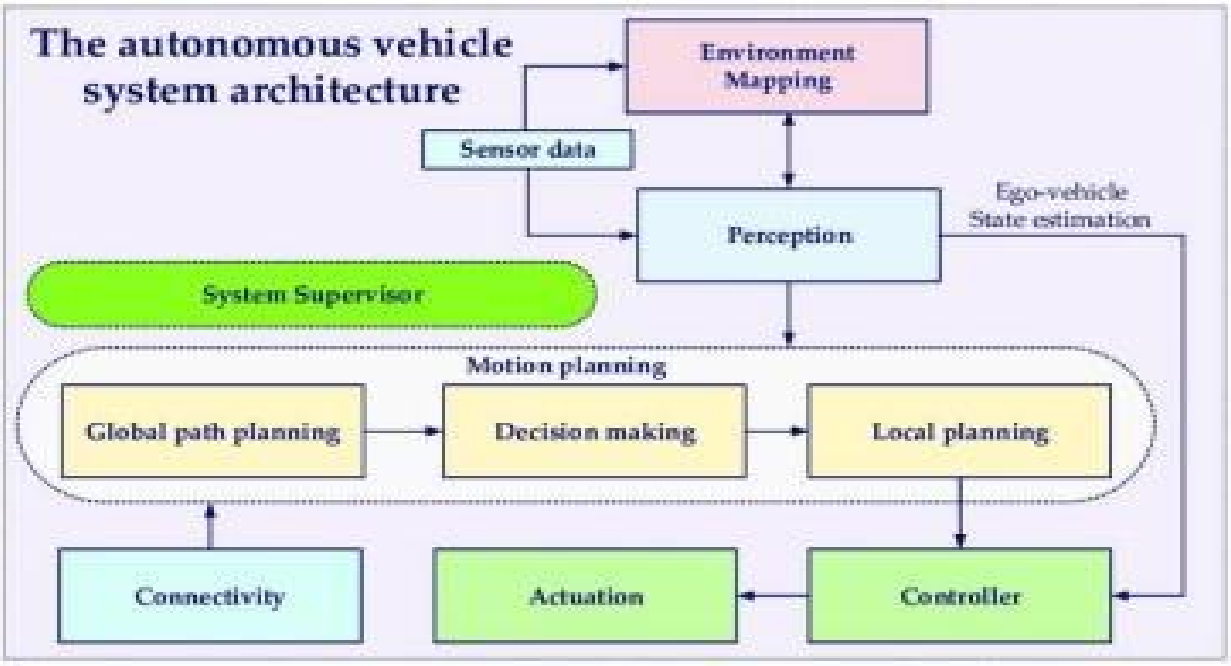


Figure 3.1: Project Architecture of DDS for Autonomous vehicle

3.2 DESCRIPTION

1. Environment Mapping:

- This module is responsible for understanding the vehicle's surroundings. It gathers data from various sensors (e.g., cameras, LiDAR, radar) to build a map of the environment.
- **Input**: Sensor data from the external environment.
- **Output**: A mapped representation of the surroundings which includes obstacles, roads, and other vehicles.

2. Perception:

- The **Perception** module processes the sensor data to detect and identify objects, track their movements, and estimate the vehicle's own position in the environment.

- **Input**: Sensor data and environment mapping.
- **Output**: Ego-vehicle state estimation (e.g., position, orientation) and detailed information about surrounding objects (e.g., other cars, pedestrians).

3. System Supervisor:

- This is the central controller overseeing the entire system's operations. It ensures that the system's goals are met by coordinating different modules.
- **Input**: Data from **Perception** and other system modules.
- **Output**: Commands to control the behavior of the vehicle (e.g., adjust speed, change direction).

4. Motion Planning:

- **Motion planning** is divided into three sub-modules:
    - **Global Path Planning**: Generates a high-level route from the current location to the desired destination, considering traffic rules, road conditions, etc.
    - **Decision Making**: This module evaluates different options (e.g., lane change, obstacle avoidance) and selects the best course of action for the vehicle based on the current situation.
    - **Local Planning**: Develops a detailed plan for the vehicle's movement within its immediate environment, considering speed, acceleration, and nearby obstacles.
- **Input**: Ego-vehicle state estimation, global path, and sensor data.
- **Output**: A precise path the vehicle should follow for safe driving.

5. Connectivity:

- This module ensures the vehicle can communicate with external systems (e.g., cloud servers, other vehicles, or roadside infrastructure) to receive updates, traffic information, or warnings.
- **Input/Output**: Data communication between the vehicle and external entities.

6. Actuation:

- **Actuation** converts the planned decisions into physical actions, controlling the vehicle's throttle, brake, and steering to follow the desired path.
- **Input**: Movement commands from the **Local Planning** and **Motion Planning** modules.
- **Output**: Physical movement of the vehicle.

7. Controller:

- This module continuously monitors the vehicle's actual performance and adjusts the actuators to correct any deviations from the planned path.
- **Input**: Feedback from the vehicle's sensors (speed, steering angle, etc.) and the desired trajectory.
- **Output**: Fine-tuned control signals sent to the actuators to ensure smooth and safe driving.

3.3 USE CASE DIAGRAM

In the use case diagram we have basically two actors who are the user and the vehicle. The user has the rights to upload data, access to resources and train's model by Random Forest, MLP, Genetic Algorithm. Whereas the vehicle provides the data, ml model will predict the decision which is executed by the vehicle.

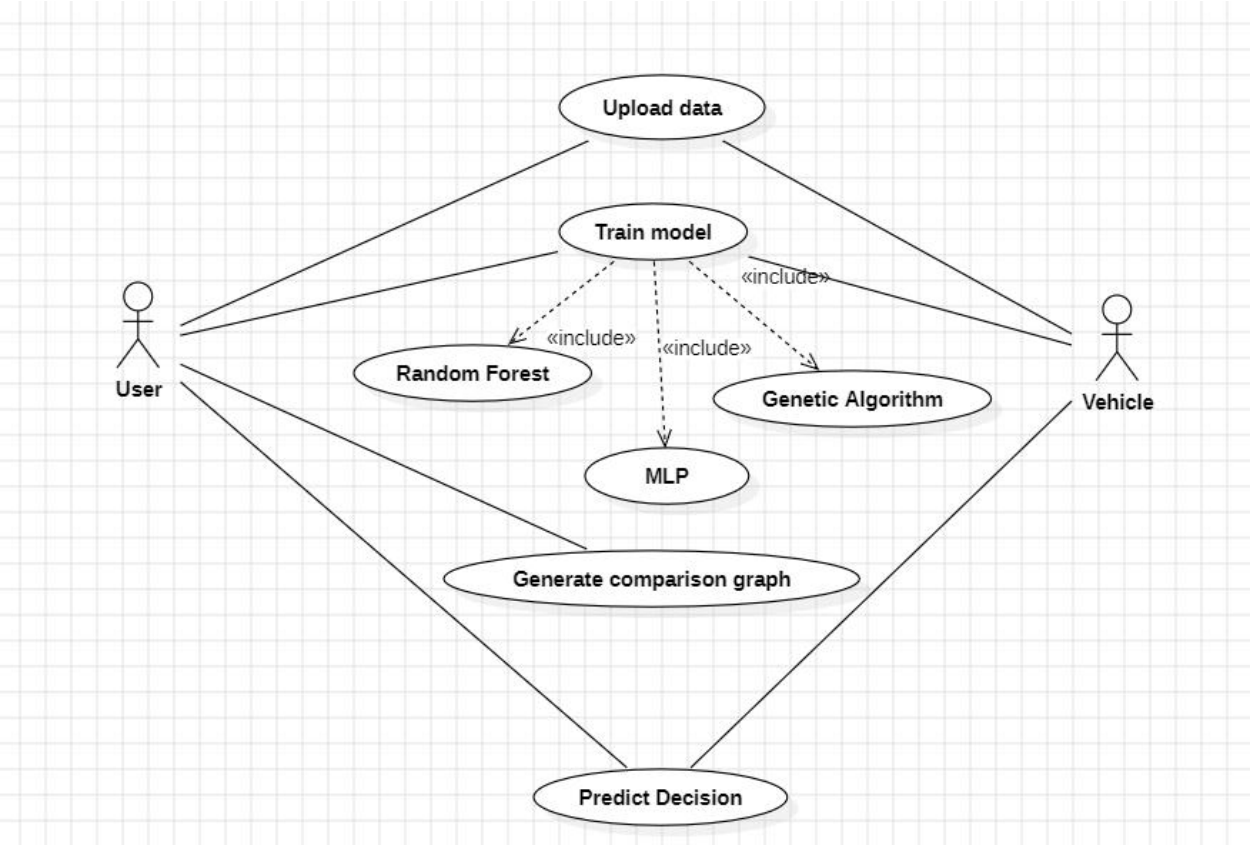


Figure 3.2: Use Case Diagram for user for DDS for Autonomous vehicle

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

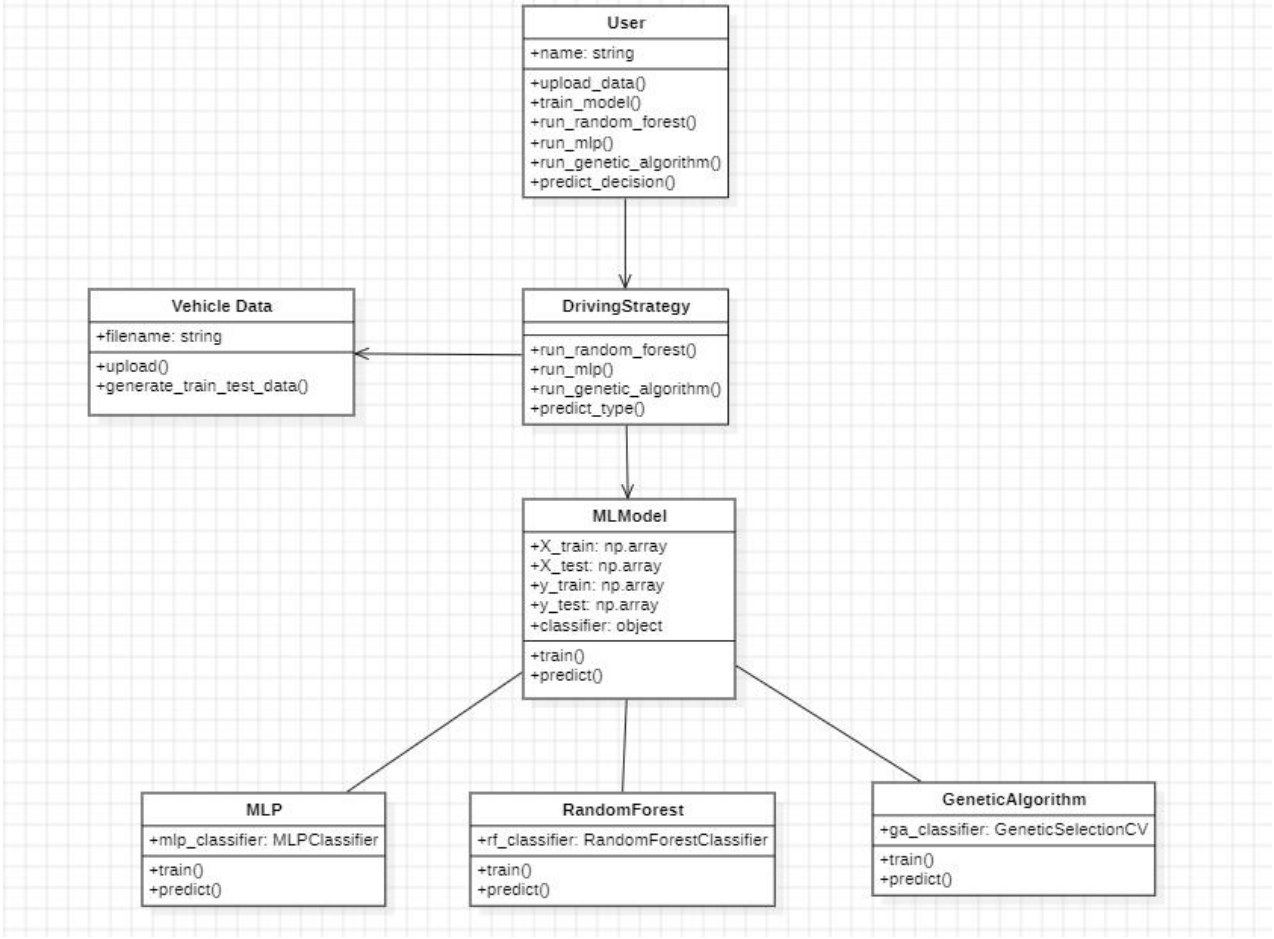


Figure 3.3: Class Diagram for user and vehicle ML Model

3.5 SEQUENCE DIAGRAM

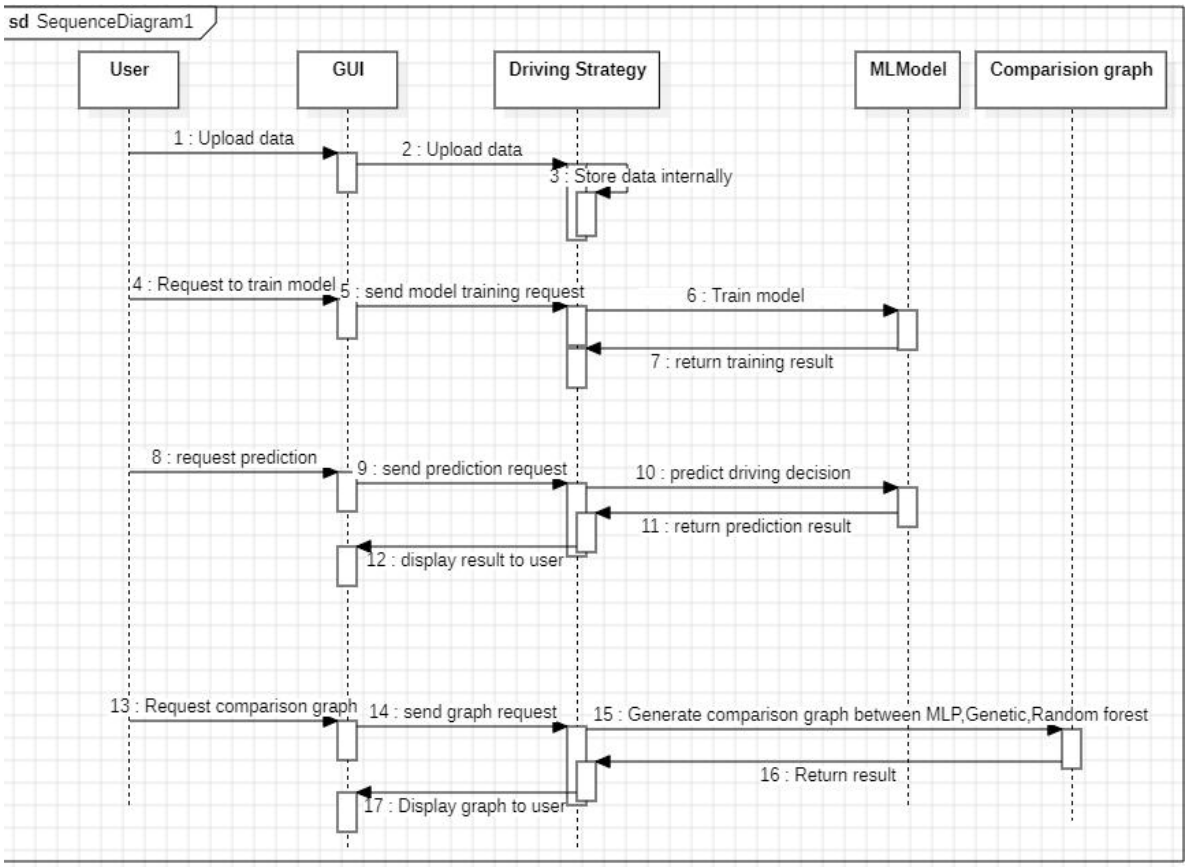


Figure 3.4: Sequence Diagram for DDS for Autonomous vehicle

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

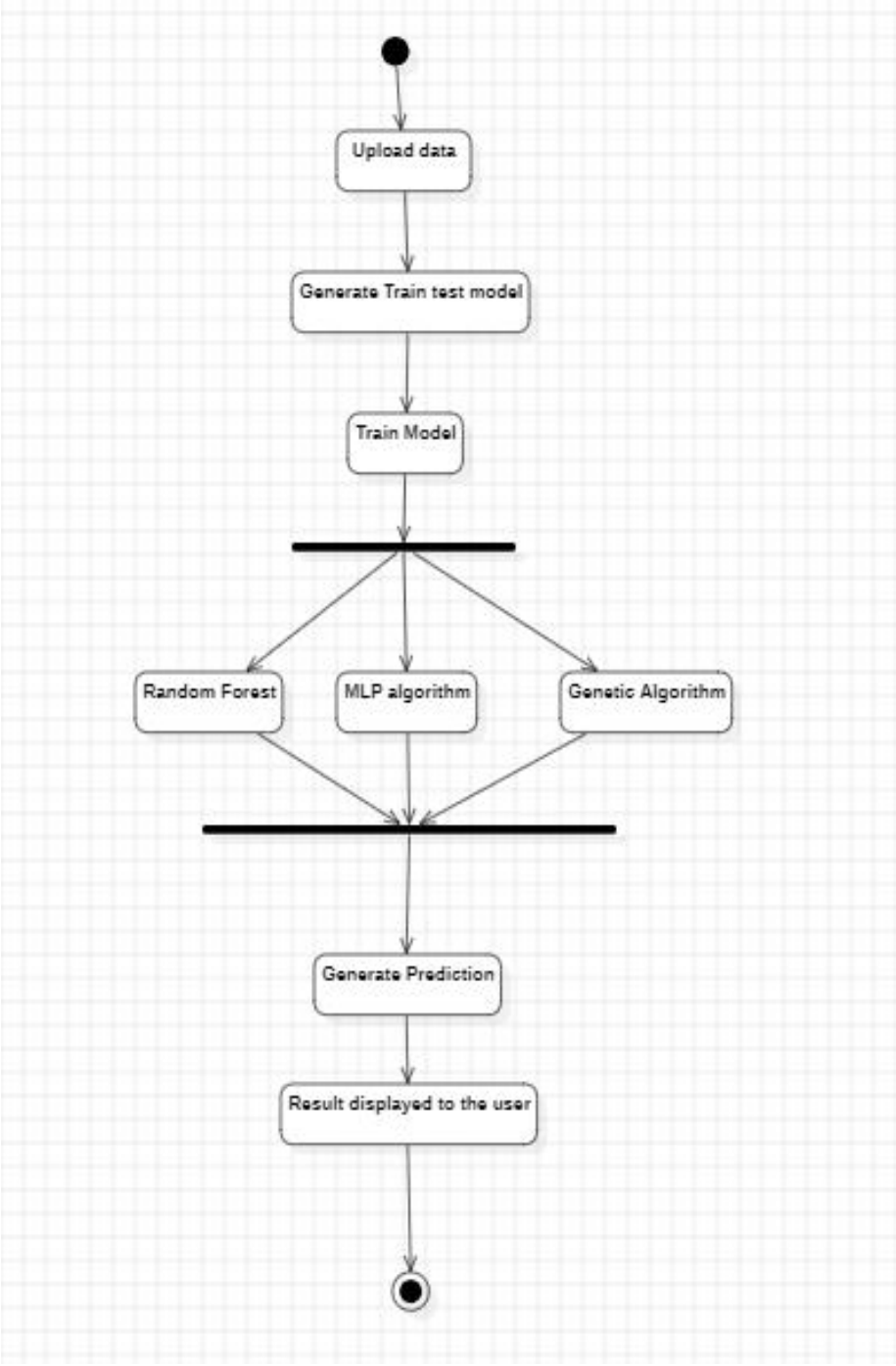


Figure 3.5: Activity Diagram for User for DDS for Autonomous vehicle

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

DDS.py:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from genetic_selection import GeneticSelectionCV

main = tkinter.Tk()
main.title("Driving Decision Strategy") #designing main screen
main.geometry("1300x1200")

global filename
global X
le = LabelEncoder()
global mlp_acc, rf_acc, dds_acc
global classifier

def upload(): #function to driving trajectory dataset
    global filename
    filename = filedialog.askopenfilename(initialdir="DrivingDataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n")

def generateTrainTestData():
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    train = pd.read_csv(filename)
    train.drop('trajectory_id', axis=1, inplace=True)
    train.drop('start_time', axis=1, inplace=True)
    train.drop('end_time', axis=1, inplace=True)
    print(train)
    train['labels'] = pd.Series(le.fit_transform(train['labels']))
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:features]
    Y = train.values[:, features]
    print(Y)
```

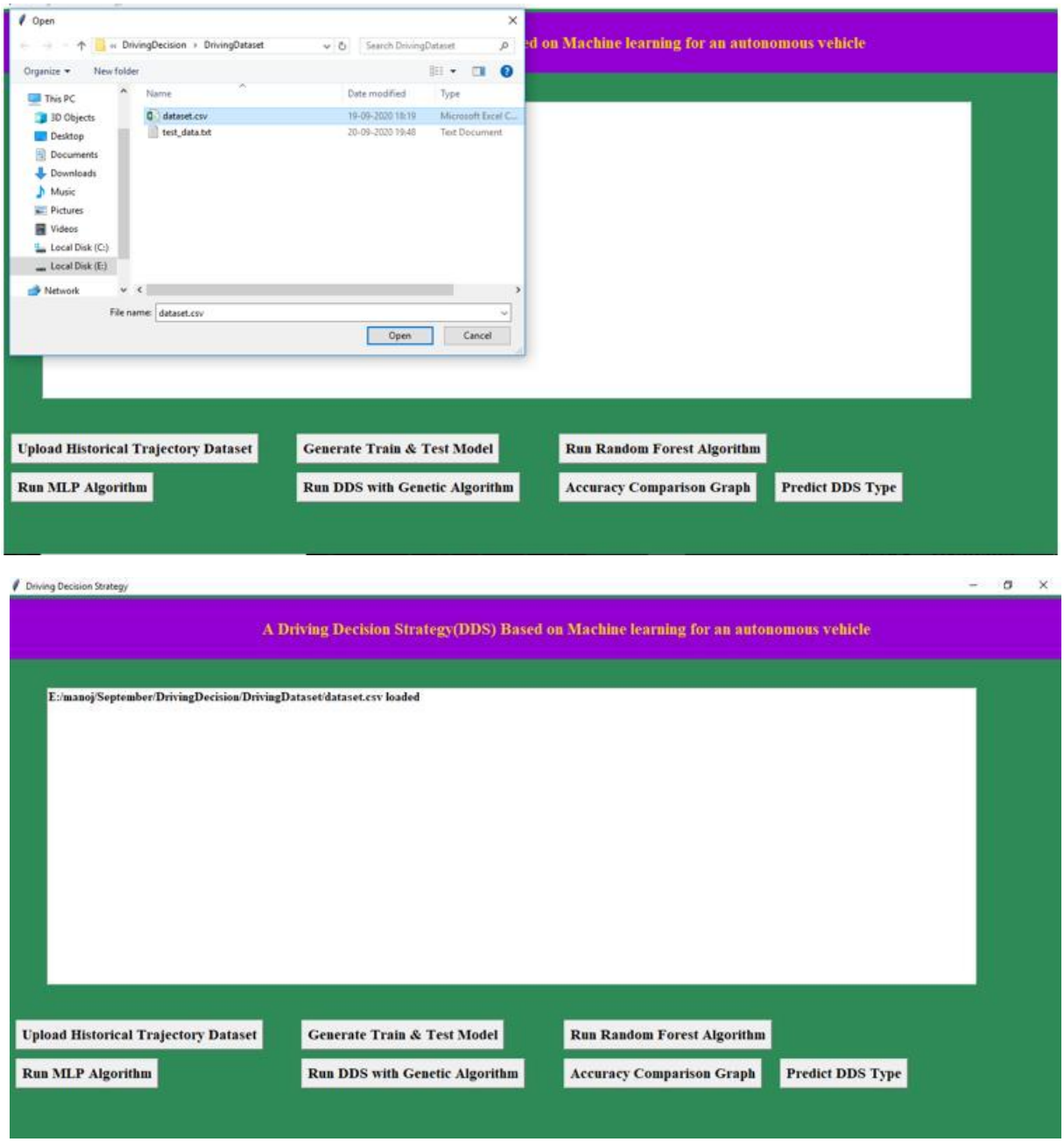
```
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)


    text.insert(END,"Dataset Length : "+str(len(X))+"\n");
    text.insert(END,"Splitted Training Length : "+str(len(X_train))+"\n");
    text.insert(END,"Splitted Test Length : "+str(len(X_test))+"\n\n");

def prediction(X_test, cls):  #prediction done here
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
    accuracy = accuracy_score(y_test,y_pred)*100
    text.insert(END,details+"\n\n")
    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
    return accuracy

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

def runMLP():
    global mlp_acc
    text.delete('1.0', END)
    cls = MLPClassifier(random_state=1, max_iter=10)
    cls.fit(X_train, y_train)
    text.insert(END,"Multilayer Perceptron Classifier (MLP) Prediction Results\n")
    prediction_data = prediction(X_test, cls)
    mlp_precision = precision_score(y_test, prediction_data,average='macro') * 100
    mlp_recall = recall_score(y_test, prediction_data,average='macro') * 100
    mlp_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    mlp_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"Multilayer Perceptron Precision : "+str(mlp_precision)+"\n")
    text.insert(END,"Multilayer Perceptron Recall : "+str(mlp_recall)+"\n")
    text.insert(END,"Multilayer Perceptron FMeasure : "+str(mlp_fmeasure)+"\n")
    text.insert(END,"Multilayer Perceptron Accuracy : "+str(mlp_acc)+"\n")

def runDDS():
    global classifier
    global dds_acc
    dds = RandomForestClassifier(n_estimators=45, random_state=42)
    selector = GeneticSelectionCV(dds,  #algorithm name
                                  cv=5,
                                  verbose=1,
```

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

def graph():
    height = [rf_acc, mlp_acc,dds_acc]
    bars = ('Random Forest Accuracy','MLP Accuracy','DDS with Genetic Algorithm Accuracy')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()

def predictType():
    filename = filedialog.askopenfilename(initialdir="DrivingDataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
    test = pd.read_csv(filename)
    test.drop('trajectory_id', axis=1, inplace=True)
    test.drop('start_time', axis=1, inplace=True)
    test.drop('end_time', axis=1, inplace=True)
    cols = test.shape[1]
    test = test.values[:, 0:cols]
    predict = classifier.predict(test)
    print(predict)
    for i in range(len(test)):
        if predict[i] == 0:
            text.insert(END,str(test[i])+" : Decision Strategy is : Lane Change\n")
        if predict[i] == 1:
            text.insert(END,str(test[i])+" : Decision Strategy is : Speed\n")
        if predict[i] == 2:
            text.insert(END,str(test[i])+" : Decision Strategy is : Steering Angle\n")


font = ('times', 16, 'bold')
title = Label(main, text='A Driving Decision Strategy(DDS) Based on Machine learning for an autonomous vehicle')
title.config(bg='darkviolet', fg='gold')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
```

```
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)

font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Historical Trajectory Dataset", command=upload)
uploadButton.place(x=10,y=550)
uploadButton.config(font=font1)

trainButton = Button(main, text="Generate Train & 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()
```

5. SCREENSHOTS

5.1 UPLOADING DATA



Screenshot 5.1: Uploading data for training the model

5.2 Creating Train Test Model

Screenshot 5.2.1 Dataset Split

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

Screenshot 5.2.2: Random Forest Algorithm Precision, recall, Accuracy

In above screen we calculated random forest accuracy, precision, recall and fmeasure and random forest got 67% prediction accuracy.

Screenshot 5.2.3 MLP Precision, recall, Accuracy

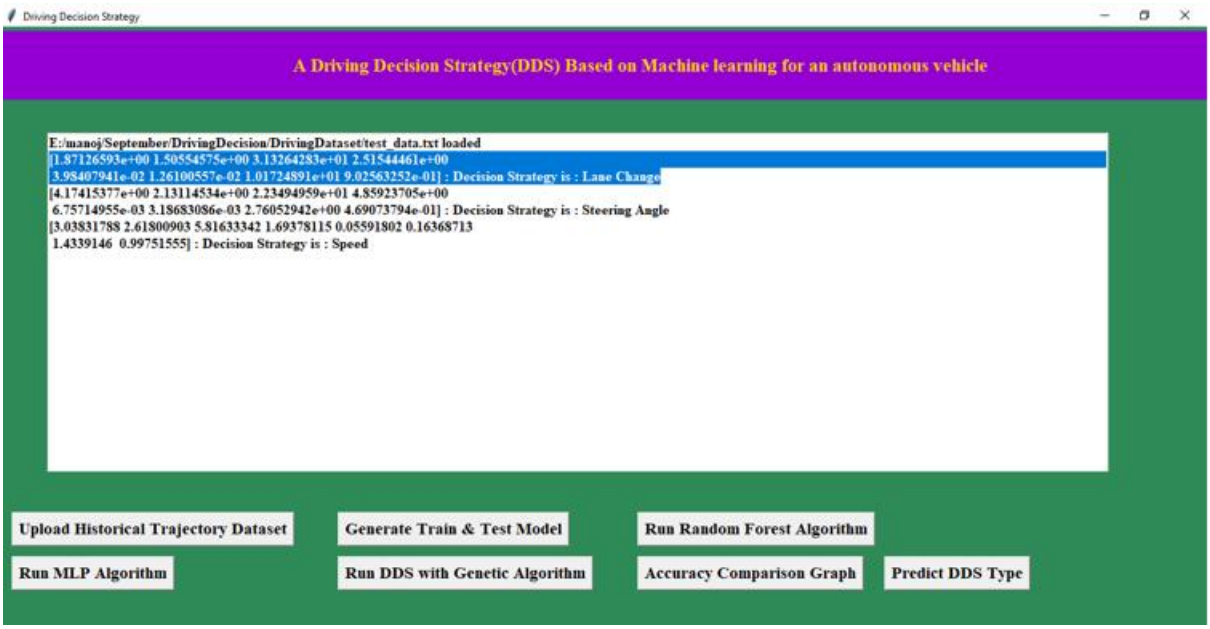
In above screen MLP got 48% prediction accuracy.

Screenshot 5.2.4: Gene values

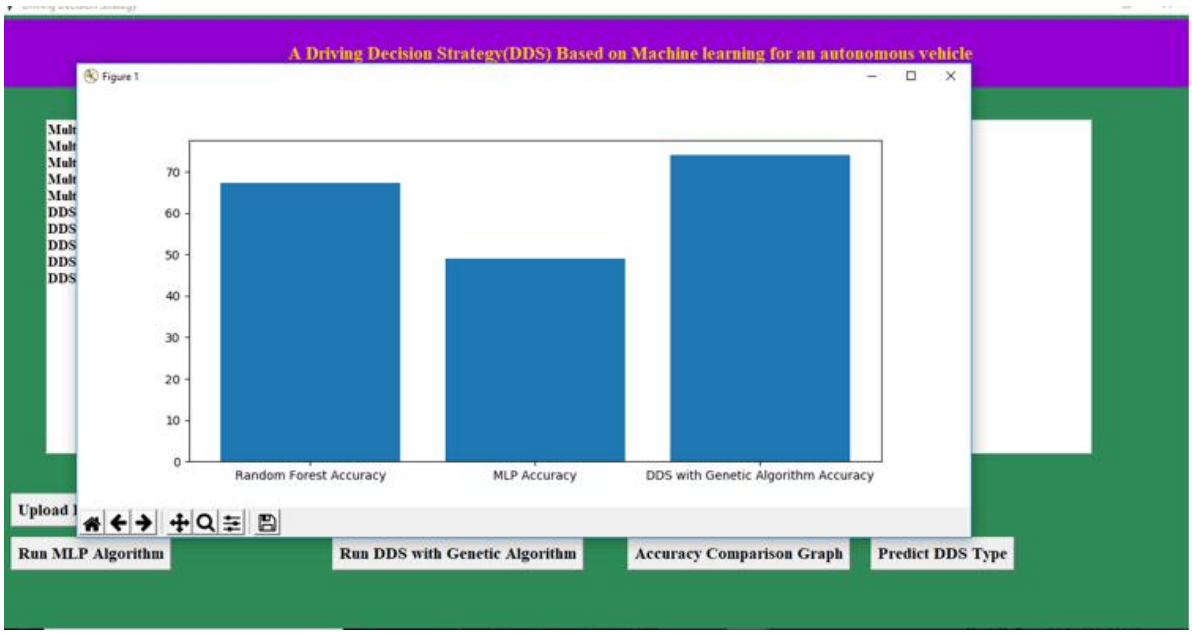
In above black console genetic algorithm starts optimal feature selection.

5.3 RESULT

In this screenshot we can see the result of the prediction



Screenshot 5.3.1: Prediction Result



Screenshot 5.3.2: Accuracy comparison graph

6. TESTING

6. TESTING

6.1 INTRODUCTION TO 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, subassemblies, 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

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

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.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

6.3.1 UPLOADING DATASET

Test case ID	Test case name	Purpose	Test Case	Output
1	Upload dataset	Upload data to predict decision	The user uploads the dataset	Uploaded successfully
2	User click's on train test model	To split the dataset into training and testing	The user splits the data into two parts one is for training and other for testing	Train Test Model successfully created.

6.3.2 PREDICTING RESULT

Test case ID	Test case name	Purpose	Test Case	Output
1	Model 1	To check the Accuracy, Precision, Recall	The model is trained using Random Forest	Accuracy, Precision, Recall are obtained
2	Model 2	To check the Accuracy, Precision, Recall	The model is trained using MLP	Accuracy, Precision, Recall are obtained
3	Model 3	To check the Accuracy, Precision, Recall	The model is trained using Genetic Algorithm	Accuracy, Precision, Recall are obtained

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

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

7.2 FUTURE SCOPE

The future scope of the Driving Decision Strategy (DDS) project is promising, with numerous avenues for advancement. Enhancing data integration from various sensors, such as LiDAR and cameras, can significantly improve the system's situational awareness. Additionally, incorporating real-time learning capabilities would allow the DDS to adapt dynamically to new driving conditions. Exploring vehicle-to-everything (V2X) communication can further enhance safety and efficiency by facilitating information sharing among vehicles and infrastructure. Overall, the DDS can evolve to address regulatory frameworks, public acceptance, and optimization for fleet management, paving the way for more reliable and effective autonomous vehicle systems.

8. BIBILOGRAPHY

8. BIBILOGRAPHY

8.1 REFERENCES

- 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.
- 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, pp746-256, 2008.
- Ning Y e, Yingya Zhang, Ruchuan Wang, Reza Malekian, "Vehicle trajectory prediction based on Hidden Markov Model, " The KSII Transactions on Internet and Information Systems, Vol. IO, No. 7, 2017.
- 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