JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Driving Decision Strategy (DDS) Based on Machine learning for an autonomous vehicle

Submitted by
KILLI JAYASRI
Under the Esteemed Guidance of

K. RATNAREDDY
ASSISTANT PROFESSOR

ABSTRACT

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

1. INTRODUCTION

1.1 BACKGROUND

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

proposed DDS analyzes them to determine the best driving strategy by using a Genetic algorithm stored in the Cloud.

1.2 Motivation (problem statement)

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 vehicle.

1.3 Existing system

K-NN, RF, SVM, and Bayes models are existing methods although studies have been done in the medical field with 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 the Hidden Markov Model (HMM) by the historical data. Finally, it proposes a new algorithm for vehicle trajectory prediction based on the hidden Markov model of double layers of hidden states and predicts the nearest neighbor unit of location information of the next k stages.

●.3.1 Overview of an existing system and its disadvantages

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

The system typically includes several components, such as data collection and pre-processing modules, a machine learning model, and a decision-making module. The data collection and pre-processing modules collect data from various sensors and cameras installed on the vehicle and pre-process that data to make it suitable for the machine learning model. The machine learning model uses this pre-processed data to train and learn driving patterns and make predictions about the optimal driving decisions in different situations. Finally, the decision-making module receives the predictions from the machine learning model and decides the appropriate driving action, such as steering, braking, or accelerating.

Disadvantages of the system:

Despite its many advantages, the "driving decision strategy using machine learning for an autonomous vehicle" system has several disadvantages that need to be addressed to ensure its safe and effective use. Here are some of the disadvantages:

Limited accuracy: Machine learning models are not always accurate and can make errors in their predictions. This can lead to unsafe driving decisions that can cause accidents or damage to the vehicle.

Data quality: The accuracy of machine learning models depends heavily on the quality and quantity of the data used to train them. If the data used is not representative of all possible driving scenarios, the model may not make optimal driving decisions in some situations.

Limited adaptability: The system's ability to make driving decisions is limited to the data it has been trained on. If the system encounters a new driving scenario that it has not been trained on, it may not make optimal decisions.

Cost: The development and deployment of the "driving decision strategy using machine learning for an autonomous vehicle" system can be expensive, requiring significant investment in hardware, software, and personnel.

1.4 Proposed system

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

1.4.1 Overview of the proposed system and its limitations

The proposed system for driving decision strategy using machine learning for an autonomous vehicle involves using machine learning algorithms to analyze real-time sensor data from the vehicle and make decisions on how to navigate the environment safely and efficiently.

The system would use data from sensors such as cameras, LIDAR, and radar to detect and classify objects in the environment, such as other vehicles, pedestrians, and obstacles. The machine learning algorithms would then use this data to make decisions on how to navigate the vehicle, such as when to accelerate, brake, or change lanes.

One potential limitation of this system is that it relies on the accuracy and reliability of the sensors to collect the necessary data. If the sensors are faulty or malfunctioning, it could affect the accuracy of the machine-learning algorithms and lead to unsafe driving decisions.

Another limitation is that the machine learning algorithms used in the system would need to be trained on a large and diverse dataset of real-world driving scenarios to ensure that they are making safe and effective decisions in

all possible situations. This could be a time-consuming and challenging task, as it would require collecting and labeling a vast amount of data.

Finally, there may be ethical and legal considerations when implementing an autonomous vehicle system that uses machine learning to make driving decisions. For example, who is responsible if an accident occurs due to a decision made by the machine learning algorithm, and how can we ensure that the algorithm is making ethical decisions?

1.5 Aim of the project

To develop a machine learning-based decision strategy for autonomous vehicles that maximize safety and efficiency while navigating in diverse road environments.

1.6 Scope of the project

There is significant scope for driving decision strategy using machine learning for autonomous vehicles. 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 learn 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.

Overall, the use of machine learning in driving decision strategy for autonomous vehicles can significantly enhance the performance and capabilities of these vehicles, making them safer and more efficient for passengers and other road users.

1.7 Objectives

2 Develop machine learning algorithms to predict driving decisions.

3 Enhance decision accuracy and reliability of autonomous vehicles.

4 Increase efficiency in traffic navigation.

5 Implement machine learning models to adapt to changing road conditions.

6 Reduce human error in decision-making through automated solutions.

7 Improve the overall safety of autonomous vehicles.

2. System Requirement Specifications

2.1 Purpose of the system

The purpose of a system for driving decision strategy using machine learning for an autonomous vehicle is to

enable the vehicle to make informed decisions based on the data it receives from its sensors and other sources in real time. The system uses machine learning algorithms to process and analyze the data, identify patterns, and make predictions about what actions the vehicle should take next.

The goal of such a system is to ensure safe and efficient driving while maximizing the vehicle's ability to adapt to changing conditions on the road. This requires the system to be able to learn from past experiences and use that knowledge to make better decisions in the future. The system must also be able to respond quickly and accurately to unexpected situations, such as obstacles in the road or other vehicles.

2.2 Feasibility analysis

The possibility of the project is analyzed during this part and the business proposal is placed forth with an awfully general arrangement for the project and a few value estimates. Throughout system analysis, the FEASIBILITY study of the projected system is to be distributed. This can be done to make sure that the projected system isn't a burden to the corporate. For risk analysis, some understanding of the key needs of the system is important.

- Data Availability and Quality: The availability of large and high-quality data sets is crucial for developing machine learning algorithms. In the case of autonomous vehicles, this would include data on road conditions, traffic patterns, pedestrian behavior, and weather conditions, among other factors. The feasibility of this project would depend on the availability and quality of such data.
- Technical Expertise: Developing and implementing a machine learning-based driving decision strategy would require a high level of technical expertise in machine learning, software engineering, and autonomous vehicle design. The feasibility of this project would depend on the availability of such expertise within the team or organization.
- Regulatory Environment: The development and deployment of autonomous vehicles are subject to regulations and standards set by government agencies. The feasibility of this project would depend on the regulatory environment in which the autonomous vehicle will be deployed.

2.3 Hardware requirements

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

- **Operating system : windows, linux**
- **Processor : minimum intel i3**
- **Ram : minimum 4 gb**
- **Hard disk : minimum 250gb**

2.4 Software requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Python idel 3.7 version**
- **Anaconda 3.7**
- **Jupiter**
- **Google colab**

2.5 Functional requirements

- **Data Collection:** The system must be able to collect data from various sensors on the autonomous vehicle, such as cameras, radar, and LIDAR, as well as external sources such as GPS and traffic reports.
- **Data Pre-processing:** The system must pre-process the collected data to remove any noise, errors, or outliers, and to extract relevant features for decision-making.
- **Model Selection:** The system must select appropriate machine learning models to analyse the pre-processed data and make decisions.
- **Model Training:** The system must train the selected machine learning models on a large dataset of labelled examples to optimize their accuracy and performance.
- **Real-Time Inference:** The system must be able to perform real-time inference on the pre-processed data using the trained machine learning models to make decisions for the autonomous vehicle.
- **Decision Strategy:** The system must determine a decision strategy based on the results of the machine learning models, such as accelerating, braking, turning, or changing lanes.
- **Testing and Validation:** The system must be thoroughly tested and validated in different scenarios and environments to ensure its reliability and performance.

2.6 Non-functional requirements

- **Reliability:** The system must be highly reliable and available, with minimal downtime or system failures.
- **Scalability:** The system must be able to handle large amounts of data and traffic, and must be scalable to accommodate future growth.
- **Performance:** The system must be fast and responsive, with low latency and high throughput, to ensure real-time decision-making.
- **Security:** The system must be secure and protected against unauthorized access, data breaches, or cyber-attacks.
- **Adaptability:** The system must be able to adapt to changing road conditions, traffic patterns, and weather conditions, and must be able to handle unexpected situations.

- **Usability:** The system must be user-friendly and easy to use, with a clear and intuitive interface for the human operator.
- **Compliance:** The system must comply with all applicable regulations and standards, such as safety regulations, privacy regulations, and ethical standards.
- **Cost:** The system must be cost-effective and efficient, with minimal hardware and software requirements, to ensure affordability and sustainability.

3. About the Software

What is python?

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

History of Python: -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatics). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatics (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Advantages of Python over Other Languages

- **Less Coding**

Almost all of the tasks done in Python require less coding than the same task is done in other languages. Python also has awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

- **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

- **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac, or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning**,** automate things, do web scraping, and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

- **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

- **Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle**.** The reason it is not so famous despite the existence of Brython is that it isn't that secure.

4. System Analysis and Requirements documentation

4.1 Overview

Developing a driving decision strategy using machine learning for an autonomous vehicle involves creating a set of algorithms and models that can accurately analyze and interpret data from various sources, including sensors and cameras, to make decisions about how the vehicle should operate in different scenarios.

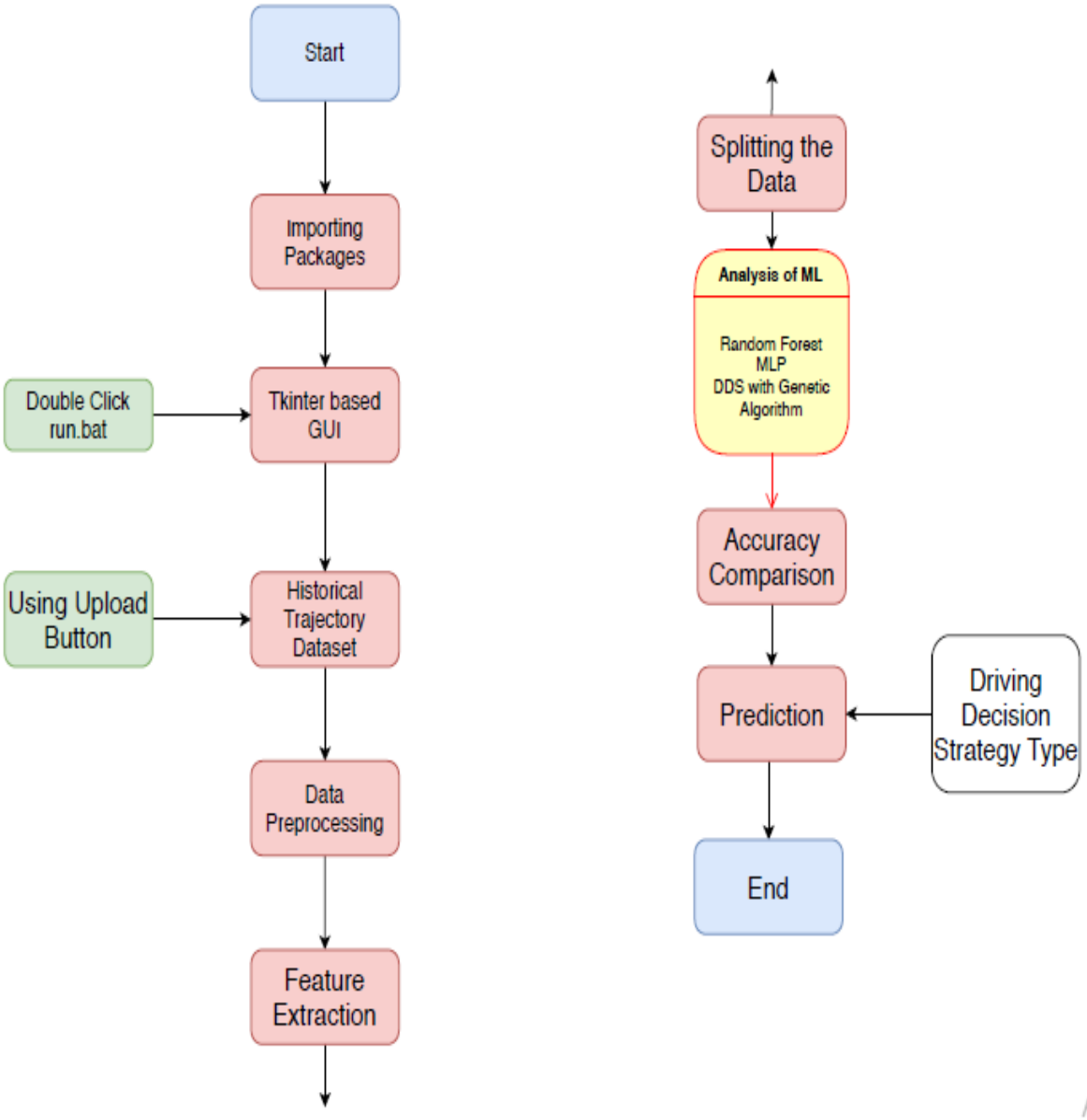
The strategy should be designed to handle a wide range of driving situations, such as navigating through traffic,

making turns, and avoiding obstacles. The machine learning algorithms should be trained on large datasets to learn patterns and make accurate predictions about how the vehicle should respond in different situations.

The strategy should also incorporate real-time feedback mechanisms that can adjust the decision-making process based on changing conditions, such as weather or traffic congestion. The system should be tested extensively in simulations and real-world environments to ensure that it is safe and reliable.

Overall, developing a driving decision strategy using machine learning for an autonomous vehicle requires a combination of expertise in machine learning, robotics, and automotive engineering, as well as a deep understanding of the challenges and complexities of driving in different environments.

4.2 Proposed System architecture/System Flow Chart



4.3 Modules

- Upload Historical Trajectory Dataset: We gather all the sensor data from the kaggale website and upload to the proposed model
- Generate Train & Test Model: We have to preprocess the gathered data and then we have to split the data into two parts training data with 80% and test data with 20%
- Run Random Forest Algorithm: we have to train the RF with train data and test the RF with test data to get best result from the algorithm
- Run MLP Algorithm: we have to train the MLP with train data and test the MLP with test data to get best result from the algorithm
- Run DDS with Genetic Algorithm : we have to train the DDS with Genetic Algorithm with train data and test the DDS with Genetic Algorithm with test data to get best result from the algorithm
- Accuracy Comparison Graph: we will find the best algorithm with highest accuracy in the form of graph
- Predict DDS Type: Enter the test data to predict the direction of a car using the DDS with Genetic Algorithm.

4.4 UML Diagrams

UML remains for Unified Modeling Language. UML is an institutionalized broadly useful displaying dialect in the field of protest situated programming designing. The standard is overseen, and was made by the Object Management Group. The objective is for UML to end up a typical dialect for making models of protest arranged PC programming. In its present shape UML is contained two noteworthy segments: a Meta-display and a documentation. Later on, some type of technique or process may likewise be added to; or connected with UML.

- The Unified Modeling Language is a standard dialect for indicating, Visualization, Constructing and recording the antiques of programming framework, and additionally for business displaying and other non-programming frameworks.

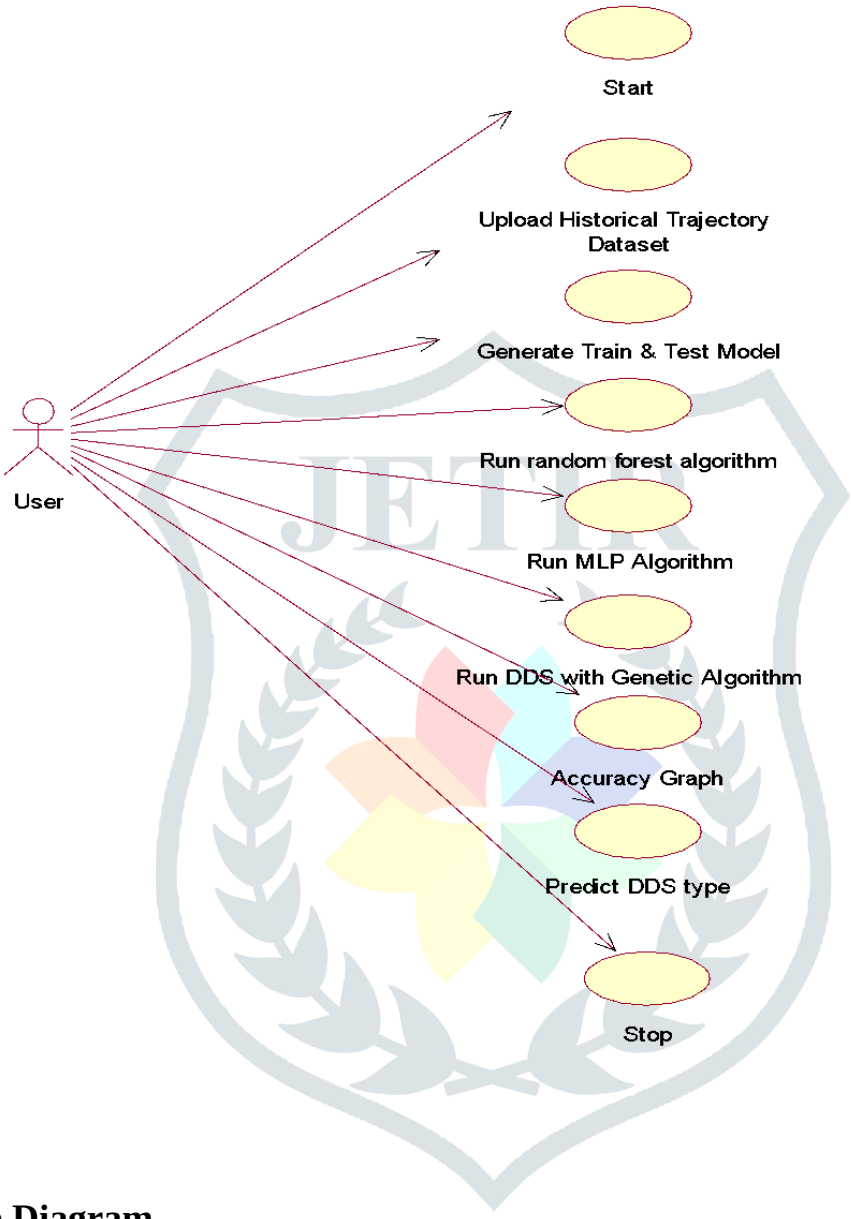
- The UML speaks to an accumulation of best building rehearses that have demonstrated effective in the displaying of vast and complex frameworks.

- The UML is an imperative piece of creating articles arranged programming and the product improvement prepare. The UML utilizes for the most part graphical documentations to express the plan of programming tasks.

4.4.1 Use case diagram

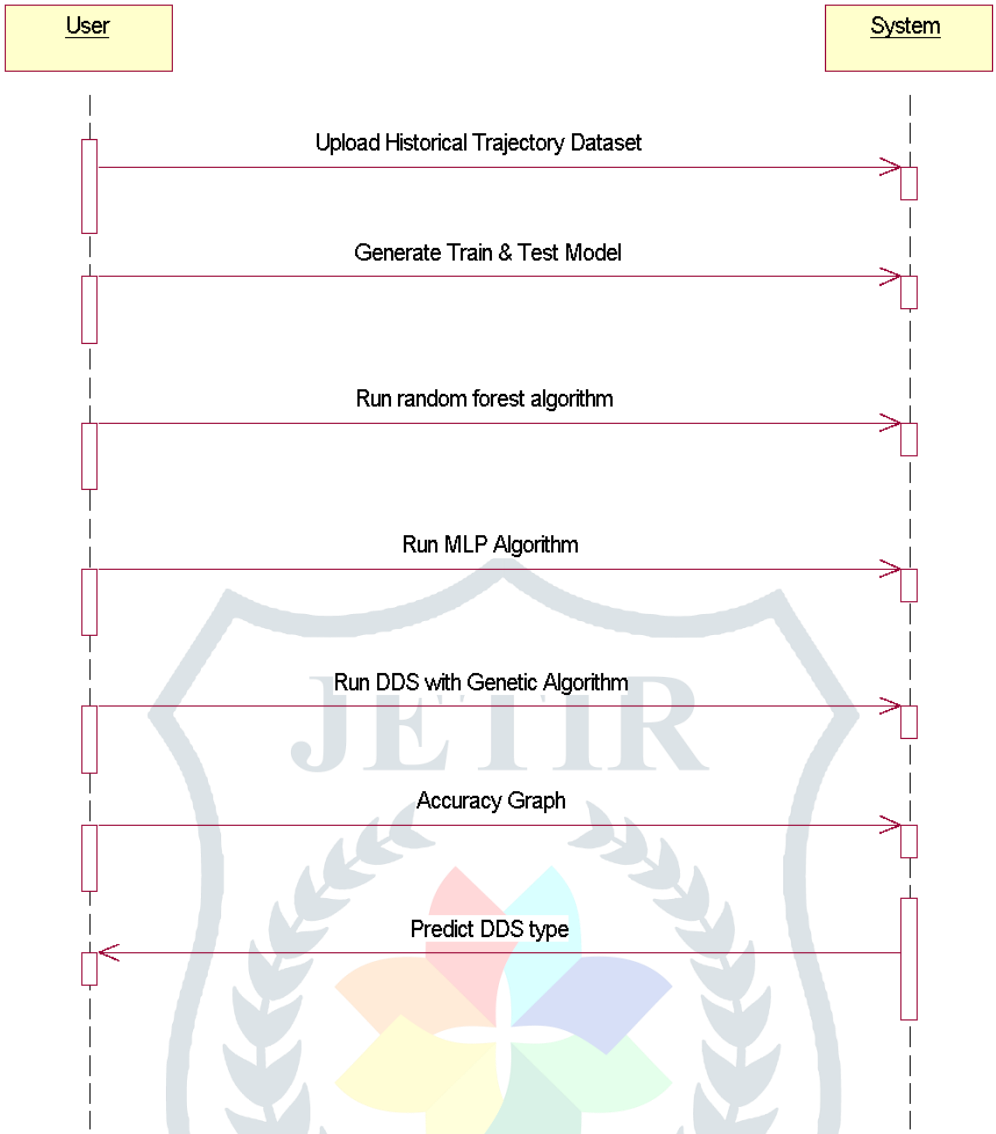
A use case diagram within the unified modeling language (UML) may be a kind of activity diagram outlined by and created from a use-case analysis. Its purpose is to gift a graphical summary of the practicality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The most purpose of a use case diagram is to indicate what system functions area unit performed that actor. Roles of the actors within the system is represented.



4.4.2 Sequence Diagram

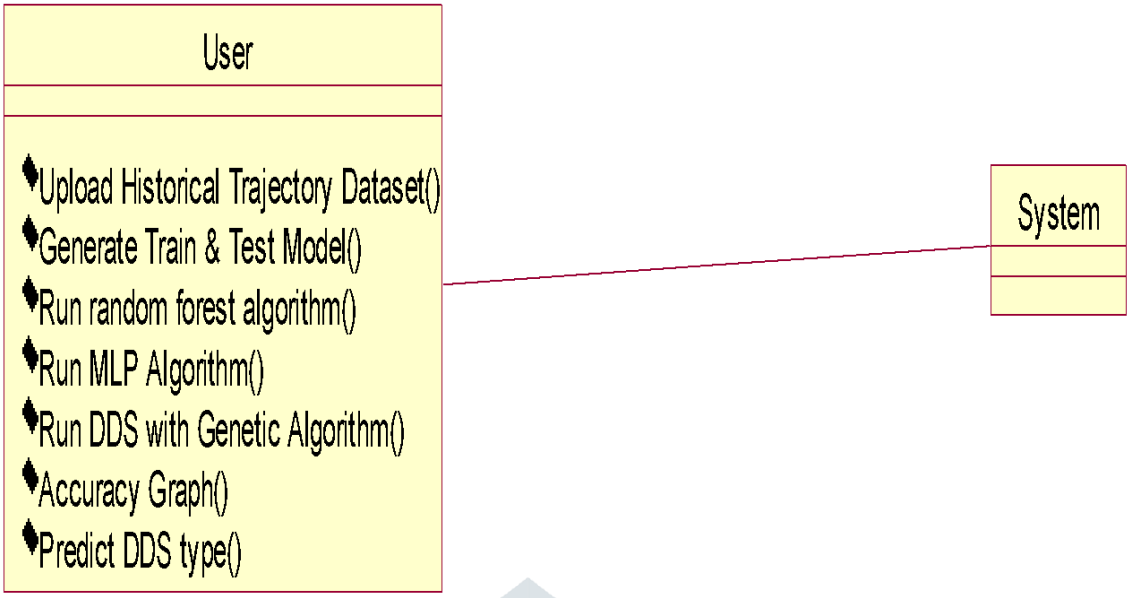
A sequence diagram in Unified Modeling Language (UML) may be a quite interaction diagram that shows however processes operate with each other and in what order. it's a construct of a Message Sequence Chart. Sequence diagrams ar generally known as event diagrams, event situations, and temporal order diagrams.



5. Structured Diagram

5.1 Class Diagram

In computer code engineering, a category diagram within the Unified Modeling Language (UML) may be a kind of static structure diagram that describes the structure of a system by showing the system's categories, their attributes, operations (or methods), and also the relationships among the categories. It explains that category contains data.



6. System Design and Documentation:

The system design involves breaking down the project into smaller components and defining the interactions between them. The documentation includes the technical details of the project, including algorithms, data structures, and implementation details.

The system design and documentation for the facial expression to emoji using CNN project typically include:

1. System Architecture: This outlines the overall structure of the system, including the hardware and software components and their interactions.
2. User Interface: This describes the graphical user interface that allows users to interact with the system and input facial expression images.
3. Data Flow Diagrams: This describes the flow of data through the system, from the input of facial expression images to the output of predicted emojis.
4. Sequence Diagrams: This shows the sequence of actions that take place when a user interacts with the system, from inputting a facial expression image to receiving the predicted emoji.
5. Class Diagrams: This shows the classes and objects used in the system and their relationships.
6. Algorithm Description: This provides a detailed explanation of the algorithms used in the project, including feature extraction and classification algorithms.
7. Implementation Details: This includes details about the programming languages, libraries, and frameworks used to implement the project.
8. Testing Plan: This outlines the testing strategy for the project, including the types of tests and the expected outcomes.
9. Maintenance Plan: This outlines the steps required to maintain the system and ensure its continued functionality.

The system design and documentation are essential components of the project as they provide a roadmap for the development process, aid in the understanding of the system, and ensure that the project can be maintained and updated over time.

6.1 Machine Learning Steps

The machine learning steps for the facial expression to emoji project are as follows:

1. Data Collection: Collect the dataset of facial expressions and corresponding emojis.
2. Data Preprocessing: Preprocess the dataset by cleaning, resizing, and normalizing the images. Split the dataset into training and testing sets.
3. Feature Extraction: Extract the facial features using OpenCV and ICP algorithm.
4. Model Selection: Choose a convolutional neural network (CNN) model for classification.
5. Model Training: Train the chosen CNN model on the preprocessed training dataset.
6. Model Validation: Validate the trained model on the preprocessed testing dataset to evaluate its performance.
7. Model Optimization: Optimize the CNN model by fine-tuning the hyperparameters and modifying the architecture.
8. Model Deployment: Deploy the final optimized CNN model for predicting the emojis from facial expressions.
9. User Interface: Develop a user interface that takes facial expression images as input and displays the predicted emoji as output.

7. 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()

8 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 assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing objectives: The most objective of testing is to uncover a bunch of errors, consistently and with minimum effort and time. Stating formally ,we can say, testing may be a method of corporal punishment a program with intent of finding miscalculation.

1. A productive check is one that uncovers Associate in Nursing hitherto undiscovered error.
2. A decent legal action is one that has likelihood of finding miscalculation, if it exists.
3. The check is insufficient to find probably gift errors.
4. The code additional or less confirms to the standard and reliable

Types of testing:

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 structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event-driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

Functional 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

BLACK BOX TESTING

Recording machine checking may be a code testing techniques during which practicality of the code below test (SUT) is tested while not staring at the interior code structure, implementation details and data of internal ways of the code .This type of testing is predicated entirely on the code needs and specifications .In recording machine Testing we have a tendency to simply concentrate on inputs and output of the package while not bothering concerning internal data of the code program. The on top of recording machine will be any package you wish to check. For example, Associate in Nursing software like Windows, a web site like Google ,a information like Oracle or maybe your own custom application. Under recording machine testing, you can check these applications by simply that specialize in the inputs and outputs while not knowing their internal code implementation.

Types of Black Box Testing

There are many varieties of recording machine testing however following ar the outstanding ones.

• **Functional testing:** This recording machine testing kind is said to purposeful needs of a system; it's done by code testers.

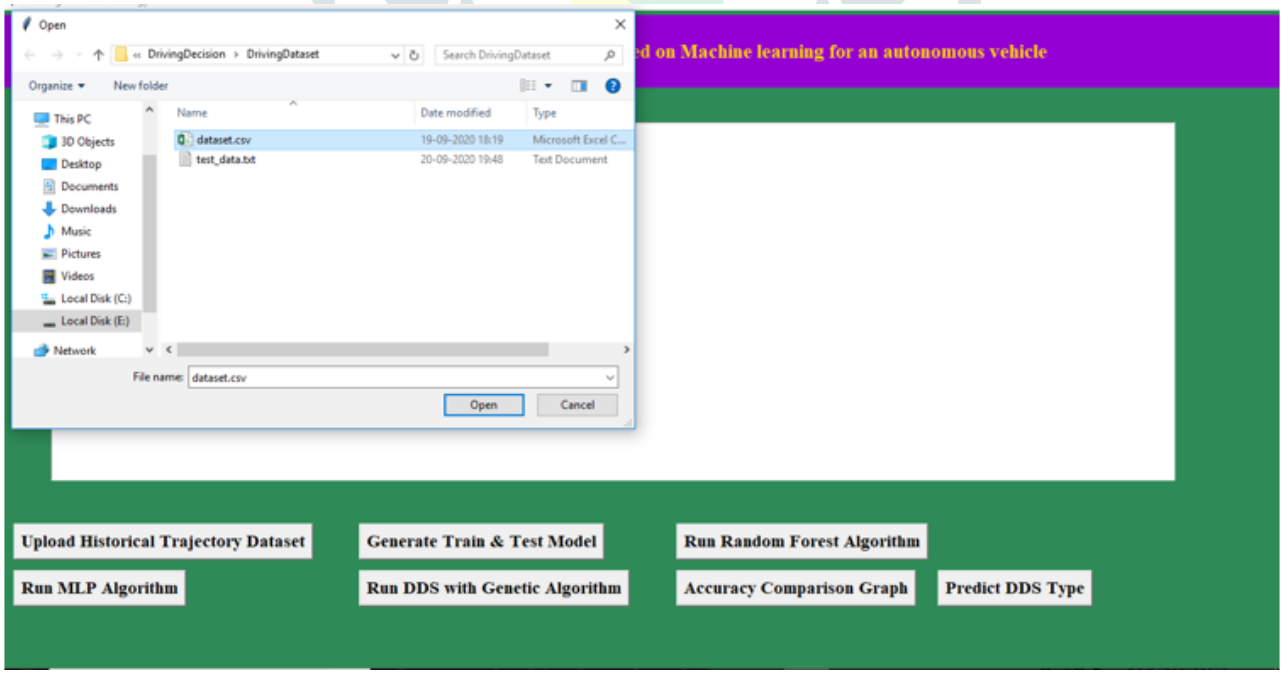
• **Non-Functional testing:** This sort of recording machine testing isn't associated with testing of a selected practicality, however non-functional needs like performance, measurability, usability.

• **Regression testing:** Regression testing is completed once code fixes, upgrades or the other system maintenance to visualize the new code has not affected the prevailing code.

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	If it doesn’t	We cannot	The application	High	High

		environment for properly loading the Application.	load.	access the application	is running successfully .		
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

9. Screenshots



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

Upload Historical Trajectory Dataset

Generate Train & Test Model

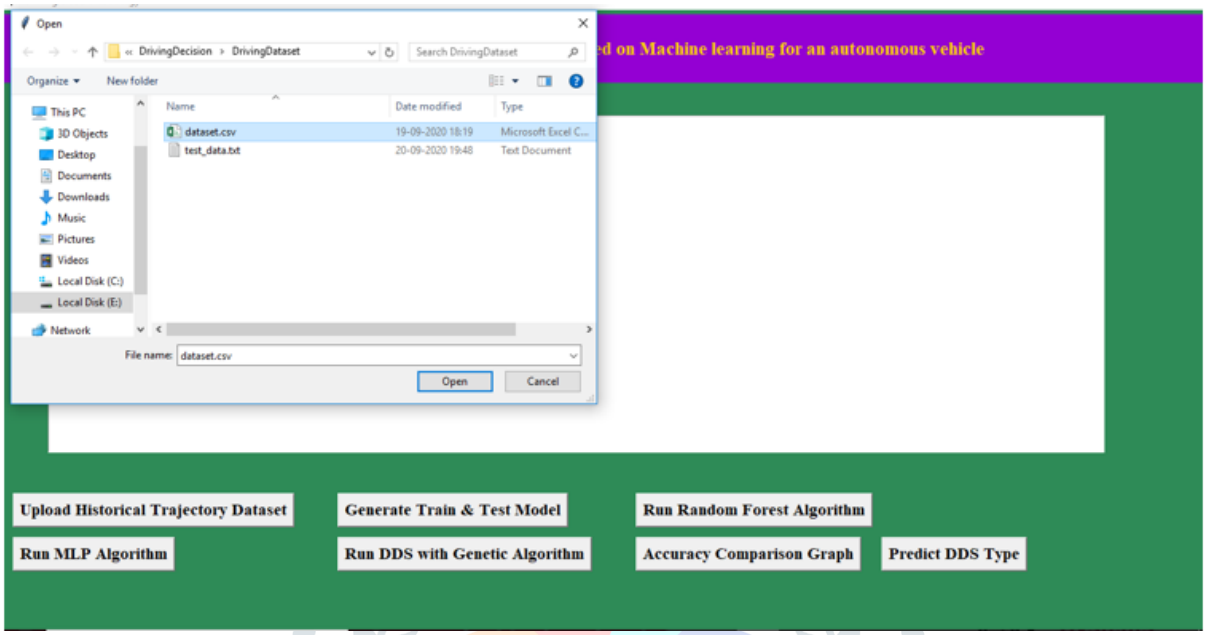
Run random forest algorithm

Run MLP Algorithm

Run DDS with Genetic Algorithm

Accuracy Graph

Predict DDS type



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

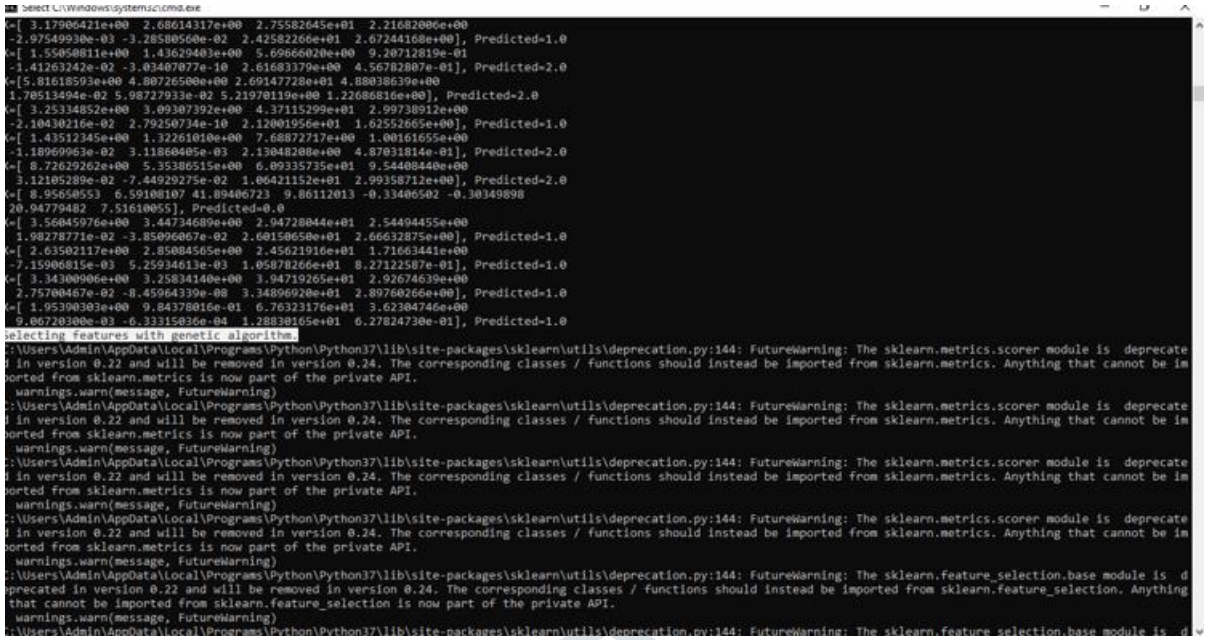
In above screen dataset is loaded and now click on ‘Generate Train & Test Model’ button to read dataset and to split dataset into train and test part to generate machine learning train model

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

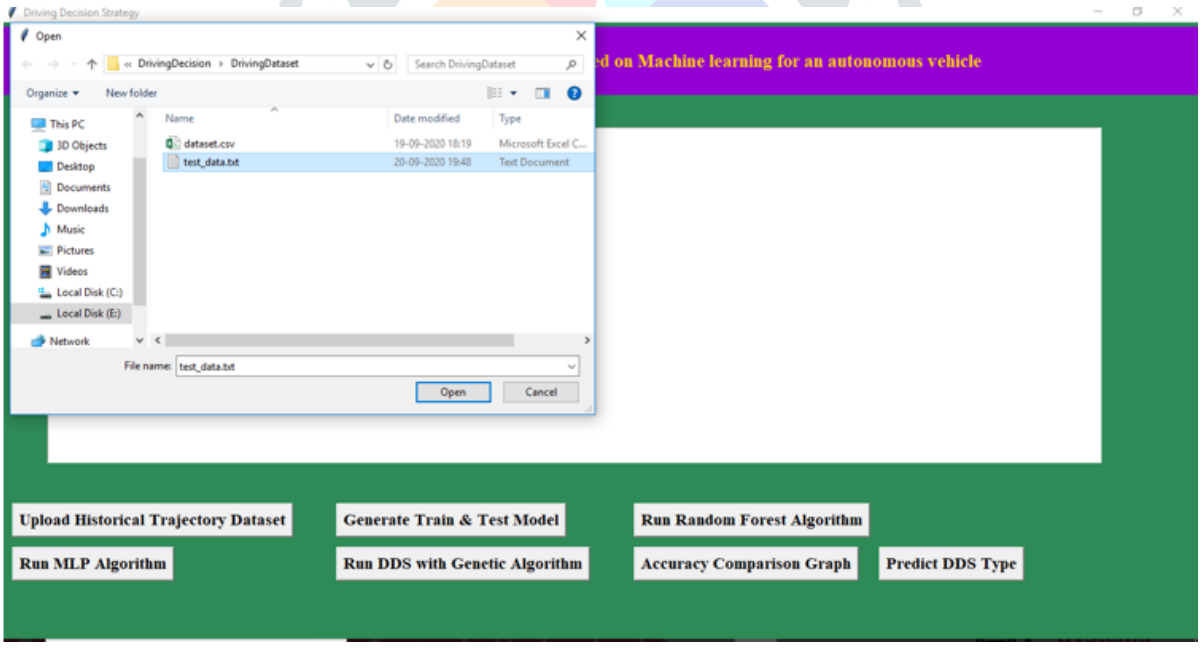
In above screen we can see genetic algorithm code used in DDS algorithm and now click on ‘Run DDS with Genetic Algorithm’ button to train DDS and to calculate its prediction accuracy



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.

10. 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 Desoto 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

11. Future Scope

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 components of the visualization...

12. Bibliography/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, pp.246-256, 2008.
- 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.
- 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