



A Driving Decision Strategy Based on Machine Learning for Autonomous Vehicle

Gunamani Jena^{1*}, Chandra Mouli VSA², Devram Yadav³, Sishir Bohara⁴,
Rupali Adhikari⁵, Pavan Lamichane⁶

Abstract

An autonomous vehicle's current driving strategy is determined solely by factors external to the vehicle (pedestrians, road conditions, etc.). This paper proposes "A Driving Decision Strategy(DDS) Based on Machine Learning for an Autonomous Vehicle," which determines the optimal strategy of an autonomous vehicle by analysing both external and internal factors of the vehicle. It is a solution to this problem (consumable conditions, RPM levels etc.). Autonomous vehicles can learn a genetic algorithm from sensor data stored in the cloud and use it to find the best driving strategy. Using MLP and RF neural network models, this paper tested the validity of the DDS.

Key Words: Genetic Algorithm, DDS, MLP

DOI Number: 10.14704/nq.2022.20.8.NQ44104

NeuroQuantology 2022; 20(8):963-969

I. Introduction

The fourth stage of self-driving cars is currently being developed by global corporations. It is possible to classify self-driving cars into three categories based on how they operate: recognition, judgement, and control. Various sensors in vehicles, such as GPS, camera, and radar, are used to gather information about the surrounding environment. Based on the gathered data, the judgement step determines the best course of action. Next, the conditions of the vehicle's placement are identified and analysed, with the goal of determining the best driving strategies to suit the driving environment and the objectives. The control step establishes the parameters of the drive, such as speed, direction, and so on, and the vehicle then takes off. It is possible for a self-driving car to accomplish a variety of tasks en route to its destination without the assistance of a human driver.

In contrast, as self-driving car performance improves, the number of sensors that can identify and process data is growing. The in-vehicle overload can be caused by an increase in the number of these sensors. Sensor data is processed by in-vehicle computers in self-driving cars. Because of

information overload, as the amount of computed data grows, it may slow down decision-making and control. The vehicle's stability may be jeopardised if these issues are not addressed. Some studies have developed hardware that can perform deep learning operations inside the vehicle, while others use the cloud to compute the sensor data from the vehicle in order to prevent the overload. Real-time data like images and sensor data from vehicles are used in existing studies to determine how the vehicle is driving. An autonomous vehicle's in-vehicle computation is reduced by generating large amounts of driving data in the cloud and determining an optimal driving strategy that takes into account the cloud's historical data, according to this paper's Driving Decision Strategy (DDS). A Genetic algorithm is used to determine the best driving strategy for the proposed DDS.

1.1 In-Vehicle Computer Vision

Hidden Markov Model (HMM) parameters are selected from historical data by mining the double layers of hidden states in vehicle trajectories.

Corresponding author: Gunamani Jena
Address: ^{1,2}Prof CSE:BVCE, ^{3,4,5,6}CSE:BVCE
E-mail: drgjena@gmail.com¹, mouliac@yahoo.co.in²



Using a Viterbi algorithm, it identifies hidden states sequences corresponding to the recently driven trajectory. Lastly, a new algorithm for vehicle trajectory prediction based on the hidden Markov model of double layers of hidden states is proposed, and it predicts the nearest neighbour unit of location information for the following k stages..

Due to the inaccuracy and instability of conventional wastewater quality measurements, it proposes an optional ensemble extreme learning machine modelling technique. As a component model in the optional ensemble frame, an extreme learning machine algorithm provides better generalisation performance than other machine learning algorithms. Variations in simulation results can be eliminated by using an ensemble extreme learning machine model. In order to reduce computation time and improve generalisation, an optional ensemble based on a genetic algorithm is used for eliminating bad components from all ensembles.

1.2 New DDS Proposed

On the basis of a genetic algorithm, DDS (Driving Decision Strategy) was developed by the author to select the best gene values for making better decisions and predictions. Input from sensors is fed into the DDS algorithm, which then uses a genetic algorithm to find the best possible value. Existing machine learning algorithms like Random Forest and MLP are being tested against the proposed DDS with genetic algorithm (multilayer perceptron algorithm.). Propose DDS is more accurate than random forest and MLP in predicting outcomes.

1.3 Purpose

As self-driving car performance improves, so does the number of sensors needed to recognise data. The in-vehicle overload can be caused by an increase in the number of these sensors. Sensor data is processed by in-vehicle computers in self-driving cars. Because of information overload, as the amount of computed data grows, it may slow down decision-making and control. The vehicle's stability may be jeopardised if these issues are not addressed. Some studies have developed hardware that can perform deep running operations inside the vehicle, while others use the cloud to compute the sensor data from the vehicle's sensors to prevent overload. Real-time data like images and sensor data from vehicles are used in existing studies to determine how the vehicle is driving.

1.4 Problem statement

Concept for driving decision strategy is described in this paper by observing vehicle internal data such as steering and RPM level to predict various classes such as speed (steering), changing lane etc. Internal values were not considered in any of the existing techniques, despite the fact that all of them were based on external data. So, in order to get an accurate reading on the current steering and lane-changing conditions, the author is using internal data. There is a sensor on the steering wheel that collects internal data that will be stored on the cloud and then accessed by the application to determine or predict steering conditions or lane changes.

1.5 The goal

Autonomous vehicles can learn a genetic algorithm from sensor data stored in the cloud and use it to find the best driving strategy. Using MLP and RF neural network models, this paper tested the validity of the DDS. the MLP and 22% faster than using radio frequency (RF), the DDS was able to determine the RPM, speed, steering angle and lane changes in the experiment with an approximate 5% loss rate lower than the existing vehicle gateways. 964

1.6 scope of work

One of the consequences of increased use of the highway transportation system has been worsening traffic congestion [1]. Crave accidents, in particular, are becoming more common and more serious. The increased centrifugal force that occurs when the car is turning is accompanied by a blind spot in vision. As the turning radius decreases and the lateral sliding becomes more common, collisions are more likely to occur [2]. Japan had a traffic accident rate of 41.01 percent, while China had an accident rate of 7.84 percent. According to the severity of the accident, fatal accidents of the curve account for 16.3% of all fatal accidents. Statistics show that accidents in curved areas are more likely to occur when drivers speed up when making a turn or change lanes irregularly [5]. Driver inattention or unfamiliarity with the road ahead is a common cause of car accidents while driving, and this is especially true on curvy roads [6]. In order to save lives and property, it is imperative that drivers are alerted to the impending presence of curved roads so that they can prepare accordingly, slow down, and avoid evasive manoeuvres.

DDS with genetic algorithm performance is compared to existing machine learning algorithms



like Random Forest and MLP.

II. Research Contribution (multilayer perceptron algorithm.)

Propose DDS is more accurate than random forest and MLP in predicting outcomes.

III. System Recommendations

On the basis of a genetic algorithm, DDS (Driving Decision Strategy) was developed by the author to select the best gene values for making better decisions and predictions. Input from sensors is fed into the DDS algorithm, which then uses a genetic algorithm to find the best possible value.

3.1 Algorithms

Nine machine learning methods are employed in this study (Random forest, MLP, Genetic algorithm)

3.1.1 Algorithm for generating random trees

As a result, this model is comprised of three distinct concepts: randomising the training data used to build trees, selecting some subsets of features to split nodes, and considering only a subset of all features for each simple decision tree. In a random forest, each tree receives training data from a random subset of the data points.

At least three layers of nodes: an input layer, a hidden layer, and an output layer are required for an MLP to function. A neuron uses a nonlinear activation function, with the exception of the input nodes. For training, MLP employs a supervised learning method known as back propagation..

3.1.2 Genetic Algorithm

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

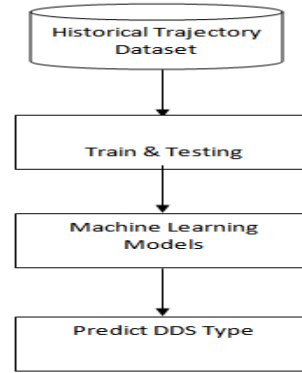
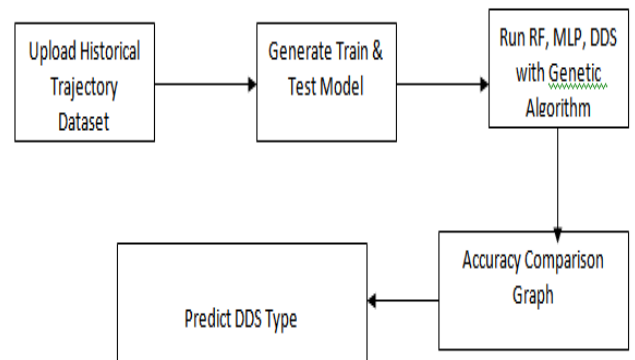


Fig-1: Architecture/Framework

3.2 Algorithm and Process Design



965

Fig.2. Process Design

IV. Data collection

Because we lack sensors to collect data, we must rely on historical vehicle trajectory datasets to carry out this project. If the driver slows down the vehicle, a sensor value with the label "lane changing" appears in the dataset. Similarly, we have different dataset classes based on values. After training on the dataset, a machine learning algorithm can be used to predict a class for a given piece of data.

The details of the dataset 'DrivingDataset' are provided below.

A trajectory's identifier (along with its start and end times and various rpms and speeds) is represented by a series of labels.

20071010152332,2007-10-10T15:23:32.000000000,2007-10-10T15:32:59.000000000,2.21513818073,2.27421615004,2.85853043655,0.428624902772,-0.005093147516729999,-0.00230819670622,0.0647143832211,0.0377402391782,speed
 20071011011520,2007-10-



11T01:15:20.000000000,2007-10-
11T01:22:10.000000000,3.71181007816,3.650651
07266,6.35783373513,1.9271696164900003,-
0.016218030061,-
0.00147783417456,0.104789889519,0.093413151
55410003,speed
20080628053717,2008-06-
28T05:37:17.000000000,2008-06-
28T05:46:42.000000000,4.65889245882,3.128299
31751,13.0268086376,4.09914234541,0.0040470
3387141,0.0124246102197,2.11899984839,0.752
1915347560001,steering angle
20080628124807,2008-06-
28T12:48:07.000000000,2008-06-
28T12:57:16.000000000,1.71674094314,1.313989
45454,18.5776836518,2.18497323244,-
0.0312684175217,0.0308633583269,2.938885587
93,0.7139256777420001,steering angle
20080825044741,2008-08-
25T04:47:41.000000000,2008-08-
25T05:05:12.000000000,2.38238360506,1.537175
8264500002,20.919113327999998,2.865359735,-
0.00720368601786,-
0.000910857743471,2.01833073218,0.471527016
571,
lane change
The dataset column names are bolded in the
example above, and the dataset values are listed
beneath them. Each sensor report includes the
trajectory id, date, and time, as well as speed and
rpm information. Labels such as STEERING ANGLE
and SPEED are shown in the last column of our
dataset. We will train a machine learning algorithm
using the above dataset values and the label to
calculate accuracy. Using only sensor values and a
trained model, we can predict or determine a
driver's decision based on the results shown below,
which do not include any class labels.
Trajectory id,start time,end time,rpm average,rpm
medium,rpm max,rpm std,speed average,speed
medium,speed max,speed std
20080823105259,2008-08-
23T10:52:59.000000000,2008-08-
23T11:03:41.000000000,1.871265931,1.50554575
041,31.326428333800006,2.51544461011,0.0398
40794139,0.0126100556557,10.1724891367,0.90
256325184 \s20080821073812,
2008-08-21T07:38:12.000000000,2008-08-
21T08:30:53.000000000,4.17415377139,2.131145
34045,22.3494958748,4.85923705089,0.0067571
4954958,0.003186830858360001,2.76052942367,
0.469073794101
20080913092418,2008-09-

13T09:24:18.000000000,2008-09-
13T09:24:36.000000000,3.03831788365,2.618009
0273700003,5.81633341636,1.6937811468,0.055
9180233599,0.163687128621,1.43391460095,0.9
97515549234

It's possible to predict or determine a driver's
driving strategy by using the above-mentioned test
data with a machine learning trained model and
then applying the model's predictions to actual test
data.

Using this module, we will upload a historical
trajectory dataset to the application, and then
determine the total number of records.

Create a test and training model: Using this module,
a machine learning train model is created by reading
the dataset and dividing it into training and testing
sections.

To build a Random Forest trained model, we'll use
this module to divide our dataset into training and
testing sets. Test data will be used to calculate and
evaluate the accuracy of predictions made by a
trained model.

This module is used to run the MLP algorithm, which
divides the dataset into train and test sets before
creating an MLP-trained model. Test data will be
used to calculate and evaluate the accuracy of
predictions made by a trained model.

Fourth, put into action and results

4.1 The dataset

Because we lack sensors to collect data, we must rely
on historical vehicle trajectory datasets to carry out
this project. If the driver slows down the vehicle, a
sensor value with the label "lane changing" appears
in the dataset. Similarly, we have different dataset
classes based on values. This dataset will be used to
train a machine learning algorithm, which will then
be able to classify new data when it is fed into the
trained model.

To evaluate the F1-Score, Accuracy and Receiver
Operating Characteristics-Area, the following
metrics were considered: We use ROC-AUC metrics
to gauge how well our models are performing.
FPR=False Positive Rate must be used to evaluate
the F1-score, accuracy, precision, and recall.

To put it another way, TPR stands for True Positive
Rate.

Precision, Accuracy, and Recall are all terms used to
describe F1-score

Values are calculated in this manner, and the results
are evaluated in terms of:

The number of events that constitute a true positive
(TP) is the number of events that have been



accurately counted.
 Unneeded or incorrectly predicted events are known as false negatives (FNs).
 Incorrectly predicted number of events is known as a false-positive (FP).
 No. of events that were foreseen but not required. (TN)
 Machine learning accuracy can be evaluated using the False Positive Rate (FPR), which measures the number of false positives in the system. The following is an explanation of what it means:
 $FPR = FP / (FP + TN)$
 a measure of the accuracy of a test result
 It is a synonym for recall and is therefore defined as
 $TPR = TP / (TP + FN)$
 Simply dividing the number of correctly predicted observations by the total number of observations is an easy way to measure accuracy.
 $Accuracy = (TN + TP) / (TP + FP + TN + FN)$
 It's the ratio that accurately predicts positive observations in the original data.
 $TP / (TP + FN) = Recall$
 In order to get the most accurate results, precision is needed. In other words, this means determining the total number of software's predicted to be positive that are actually positive. Precision is calculated as follows:
 $TP / (TP + FP) = TP / (TP)$

F1-score

The F-score is a way of combining precision and recall in a machine learning model. high levels of accuracy and recall Precision and recall are defined as the <https://deepai.org/machine-learning-glossary-and-terms/harmonic-mean> of the model, and it is Precision and recall are two important properties of the model, and they are both described by the term "harmonic mean" at <https://deepai.org>. The F-score is another name for it. Defintion: it is To get an F1 score, divide 2 (Precision + Recall) by 2. If you're trying to solve a classification problem, you'll want to keep an eye out for metrics that can help you determine how well you're doing. The dataset shown in the diagram contains a total of 977 records of trajectories, with 781 records (or about 80% of the dataset) used for training and 196 records (or about 20% of the dataset) used for testing. In order to train a random forest classifier and test its prediction accuracy on 20% of the test data, click on the "Run Random Forest Algorithm"

button.
 We calculated the accuracy, precision, recall, and measurement of random forests, and found that random forests had an accuracy of 67% in predicting outcomes. To train the MLP model and determine its accuracy, click the "Run MLP Algorithm" button now.
 To build our proposed DDS algorithm, we used genetic algorithm code that had a prediction accuracy of 48%.
 When developing the DDS algorithm, we made use of genetic algorithms to train the software and determine the accuracy of its forecasts.
 The genetic algorithm begins with the selection of the best features.

```

Multilayer Perceptron Classifier (MLP) Prediction Results
Multilayer Perceptron Precision : 48.202468030054234
Multilayer Perceptron Recall : 45.21672771672772
Multilayer Perceptron FMeasure : 42.57765830346475
Multilayer Perceptron Accuracy : 48.97959183673469
DDS Prediction Results
DDS Precision : 72.89847337494746
DDS Recall : 72.76251526251527
DDS FMeasure : 72.73399989839531
DDS Accuracy : 73.9795918367347
    
```

Fig.3. DDS algorithm

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

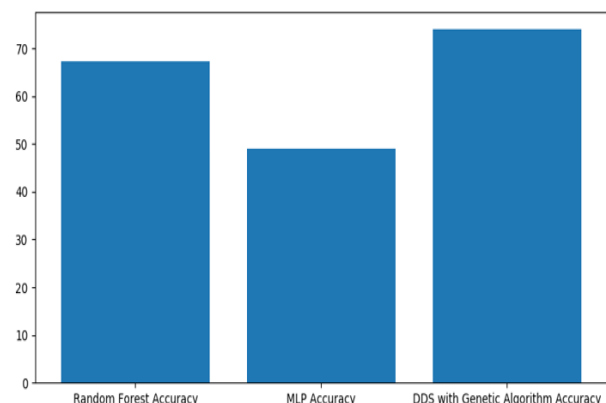


Fig.4. DDS algorithm

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. Here we uploaded 'test_data.txt' file and click on 'Open' button to predict driving decision



```
E:/manoj/September/DrivingDecision/DrivingDataset/test_data.txt loaded
[1.87126593e+00 1.50554575e+00 3.13264233e+01 2.51544461e+00
3.98407941e-02 1.26100557e-02 1.01724891e+01 9.02563252e-01] : Decision Strategy is : Lane Change
[4.17415377e+00 2.13114534e+00 2.23494959e+01 4.85923705e+00
6.75714955e-03 3.18683086e-03 2.76052942e+00 4.69073794e-01] : Decision Strategy is : Steering Angle
[3.03831788 2.61800903 5.81633342 1.69378115 0.05591802 0.16368713
1.4339146 0.99751555] : Decision Strategy is : Speed
```

Fig.4. DDS algorithm: Lane Change and steering angle'

A 'steering angle' determination was obtained for second-record values and a "vehicle speed" prediction was obtained for third-record values in the diagram above.

The End Results:

A prediction accuracy of up to 75% was obtained using Random Forest and Genetic Algorithm in the proposed DDS algorithm, and the proposed DDS algorithm is being upgraded with Genetic and XGBOOST algorithm to achieve even higher accuracy than the proposed algorithm.

The XGBOOST algorithm has the following advantages.

XGBoost is a well-known machine learning algorithm that consistently outperforms the competition. Why is XGBoost so widely used?

1) Performance and speed: It is faster than other ensemble classifiers because it was originally written in C++.

Due to the parallelizability of the core XGBoost algorithm, multi-core computers can be used to their fullest potential. It is also possible to train on very large datasets thanks to its ability to be parallelized on GPUs and across networks of computers.

When it comes to machine learning benchmark datasets, it consistently outperforms other algorithm methods.

For example, XGBoost has internal parameters for cross-validation, regularisation, user-defined objectives, missing values, tree parameters, Scikit-learn-compatible API, and so on.

Extensive Gradient Boosting (Extreme GBM) is an algorithm that uses the GBM (Gradient Boosting) framework as its foundation. It is a distributed gradient boosting library that is optimised for speed.

```
Random Forest Prediction Results
Random Forest Precision : 67.59627129540341
Random Forest Recall : 67.61446886446886
Random Forest FMeasure : 66.64871828546056
Random Forest Accuracy : 67.3469387755102

Multilayer Perceptron Classifier (MLP) Prediction Results
Multilayer Perceptron Precision : 48.202468030054234
Multilayer Perceptron Recall : 45.21672771672772
Multilayer Perceptron FMeasure : 42.57765830346475
Multilayer Perceptron Accuracy : 48.97959183673469

DDS Prediction Results
DDS Precision : 74.93150684931506
DDS Recall : 75.28235653235652
DDS FMeasure : 75.0576134682095
DDS Accuracy : 75.51020408163265

Extension DDS Prediction Results
Extension DDS with XGBoost Precision : 73.76488095238095
```

Fig.5. Random Forest

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

```
Random Forest Accuracy : 67.3469387755102

Multilayer Perceptron Classifier (MLP) Prediction Results
Multilayer Perceptron Precision : 48.202468030054234
Multilayer Perceptron Recall : 45.21672771672772
Multilayer Perceptron FMeasure : 42.57765830346475
Multilayer Perceptron Accuracy : 48.97959183673469

DDS Prediction Results
DDS Precision : 74.93150684931506
DDS Recall : 75.28235653235652
DDS FMeasure : 75.0576134682095
DDS Accuracy : 75.51020408163265

Extension DDS Prediction Results
Extension DDS with XGBoost Precision : 73.76488095238095
Extension DDS with XGBoost Recall : 75.0
Extension DDS with XGBoost FMeasure : 74.37452326468345
Extension DDS with XGBoost Accuracy : 98.46938775510205
```

Fig.6. XGBOOST

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



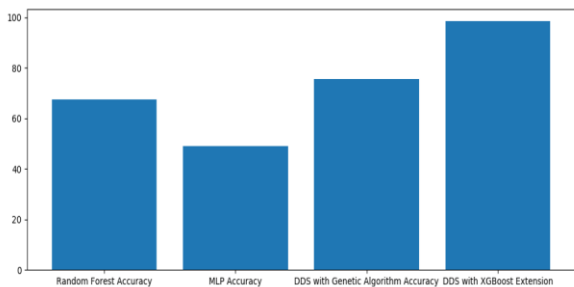


Fig.6. XGBOOST

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

```
E:\manoj\September\DrivingDecision\DrivingDataset\test_data.txt loaded
[1.87126593e+00 1.50554575e+00 3.13264283e+01 2.51544461e+00
3.98407941e-02 1.26100557e-02 1.01724891e+01 9.02563252e-01] : Decision Strategy is : Lane Change
[4.17415377e+00 2.13114534e+00 2.23494959e+01 4.85923705e+00
6.75714955e-03 3.18683086e-03 2.76052942e+00 4.69073794e-01] : Decision Strategy is : Steering Angle
[3.03831788 2.61800903 5.81633342 1.69378115 0.05591802 0.16368713
1.4339146 0.99751555] : Decision Strategy is : Speed
```

Fig.7. Prediction output

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

Conclusion

we used Random Forest and Genetic Algorithms in the proposed DDS algorithm for a prediction accuracy of 75%, and we're now upgrading it with a Genetic and XGBOOST extension, which is giving much better accuracy than the proposed algorithm does.

References

Y.N. Jeong, S.R.Son, E.H. Jeong and B.K. Lee, "An Integrated SelfDiagnosis 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

