2022

ENPM808A: Introduction to Machine Learning

Data-Driven Motion Planning using Machine Learning Algorithms

BHARADWAJ CHUKKALA

118341705

bchukkal@umd.edu

Table of Contents

1. Introduction

1.1. Background Study

- Robot Navigation
  The fundamental objective of mobile Robot Navigation is to arrive at a goal position without collision. The mobile robot is supposed to be aware of obstacles and move freely in different working scenarios. The robot arrives at the desired location by planning its path using an in-built path planning algorithm.

- Path Planning
  Motion planning, also known as path planning (also known as the navigation problem or the piano mover's problem) is a computational problem to find a sequence of valid configurations that moves the object from the source to the destination. The robot will be using a path-planning algorithm to navigate to the desired position.

- Data-Driven Model Approach
  Data-driven decision-making (sometimes abbreviated as DDDM) is the process of using data to inform your decision-making process and validate a course of action before committing to it.

1.2. Project Details

- Culmination of all concepts
  - The objective of the project is to perform path planning for a four-wheel vehicle with nonholonomic constraints, using Machine Learning
  - I shall use different ML Models to predict the outputs properly.
  - The model is being trained and tested on real-time data that has been captured using Laser scan.
  - I am using two models to perform the implementation; one is Linear Regression and the other is Deep Neural Network.
  - Machine Learning Pipeline will be followed to get the desirable result.

1.3. System Software Dependencies

- OS: Windows / Ubuntu
- IDE: VSCode
- Version control: GitHub
- Python 3.6 +
- TensorFlow
- Scikit Learn
- NumPy
- Matplotlib
- Pandas

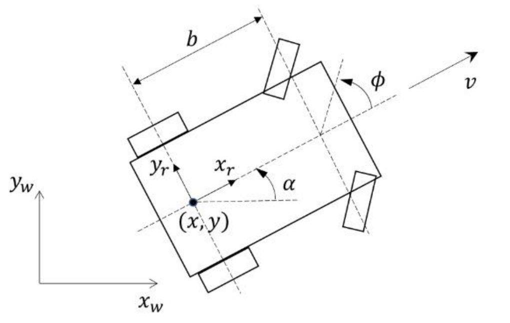
2. Project Guidelines

2.1. Description

- The LIDAR recorded data from a non-holonomic car-like robot model is given. The objective is to develop a model that takes in the control input as translation and rotational speed of the robot. The given data shows that the car has been trained to navigate in two different environments which are a corridor scenario with moving obstacles and an open box/hall environment with moving obstacles.

2.2. Problem statement

- The control inputs to this model is $u = [v, \omega]$, where $\omega \overset{\Delta}{=} \alpha$ is the rotational speed of the robot. These can be transformed to the command actions $(v, \emptyset)$, where v is the translational speed and ϕ is the steering angle obtained by $\phi = \tan^{-1}(\omega b / v)$.



$$\dot{x} = v \cos \alpha$$
$$\dot{y} = v \sin \alpha$$
$$\dot{\alpha} = \frac{v}{b} \tan \phi.$$

Figure 1: A Car like Robot

2.3. Pipeline to be Followed

- Data Collection [Training Set and Test Set]
- Data Preprocessing [Cleaning, Imputation, Normalization, Standardization]
- Model Selection
- Hyperparameter Tuning
- Regularization on tuned model
- Prediction Performance on Test Data
- Result Plots
- Generalization of model

3. Data Collection

Data collection means pooling data by scraping, capturing, and loading it from multiple sources, including offline and online sources. High volumes of data collection or data creation can be the hardest part of a machine learning project, especially at scale.

3.1. Resources

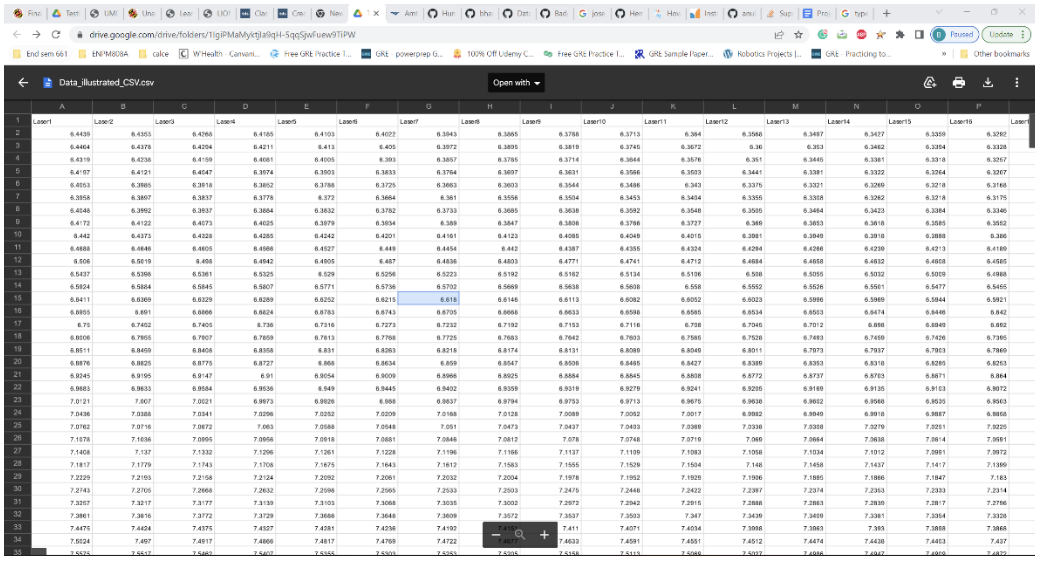
- Training Dataset
- Test Dataset

3.2. Organization of Data

➢ Each of the *.csv files has the following data:
  [Laser Range(1:1080), Final_goal_x, Final_goal_y, Final_goal_qk, Final_goal_qr, Local_goal_x, Local_goal_y, Local_goal_qk, Local_goal_qr, Robot_pos_x, Robot_pos_y, Robot_pos_qk, Robot_pos_qr, Cmd_vel_v, Cmd_vel_w]]

➢ The Data is organized in each *.csv as follows:
  - o First 1080 columns represent the laser range data,
  - o next 4 columns represent final goal information,
  - o next 4 columns represent the local goal information,
  - o next 4 columns represents robot's current position and pose information,
  - o final 2 columns represent the commanded actions

➢ A note on the generated data
  - o Two environments were considered
    - ▪ A corridor scenario with moving obstacles
    - ▪ An open box/hall environment with moving obstacles
  - o The data from these environments are in the corresponding folders.
  - o The folder "special_CSV" contains high instances of some special maneuvers such as backing up.


drive.google.com/drive/folders/1lgiPMaMyktjla9qH-5qqSjwFuew9TiPW
Data_illustrated_CSV.csv
Open with

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Laser1	Laser2	Laser3	Laser4	Laser5	Laser6	Laser7	Laser8	Laser9	Laser10	Laser11	Laser12	Laser13	Laser14	Laser15	Laser16
2	6.4439	6.4353	6.4268	6.4185	6.4103	6.4022	6.3943	6.3865	6.3788	6.3713	6.364	6.3568	6.3497	6.3427	6.3359	6.3292
3	6.4464	6.4378	6.4294	6.4211	6.413	6.405	6.3972	6.3895	6.3819	6.3745	6.3672	6.36	6.353	6.3462	6.3394	6.3328
4	6.4319	6.4238	6.4159	6.4081	6.4005	6.393	6.3857	6.3785	6.3714	6.3644	6.3576	6.351	6.3445	6.3381	6.3318	6.3257
5	6.4197	6.4121	6.4047	6.3974	6.3903	6.3833	6.3764	6.3697	6.3631	6.3566	6.3503	6.3441	6.3381	6.3322	6.3264	6.3207
6	6.4053	6.3985	6.3918	6.3852	6.3788	6.3725	6.3663	6.3603	6.3544	6.3486	6.343	6.3375	6.3321	6.3269	6.3218	6.3168
7	6.3958	6.3897	6.3837	6.3778	6.372	6.3664	6.361	6.3556	6.3504	6.3453	6.3404	6.3355	6.3308	6.3262	6.3218	6.3175
8	6.4048	6.3992	6.3937	6.3884	6.3832	6.3782	6.3733	6.3685	6.3638	6.3592	6.3548	6.3505	6.3464	6.3423	6.3384	6.3346
9	6.4172	6.4122	6.4073	6.4025	6.3979	6.3934	6.389	6.3847	6.3806	6.3766	6.3727	6.369	6.3653	6.3618	6.3585	6.3552
10	6.442	6.4373	6.4328	6.4285	6.4242	6.4201	6.4161	6.4123	6.4085	6.4049	6.4015	6.3981	6.3949	6.3918	6.3888	6.386
11	6.4688	6.4646	6.4605	6.4566	6.4527	6.449	6.4454	6.442	6.4387	6.4355	6.4324	6.4294	6.4266	6.4239	6.4213	6.4189
12	6.506	6.5019	6.498	6.4942	6.4905	6.487	6.4836	6.4803	6.4771	6.4741	6.4712	6.4684	6.4658	6.4632	6.4608	6.4585
13	6.5437	6.5398	6.5361	6.5325	6.529	6.5256	6.5223	6.5192	6.5162	6.5134	6.5106	6.508	6.5055	6.5032	6.5009	6.4988
14	6.5924	6.5884	6.5845	6.5807	6.5771	6.5736	6.5702	6.5669	6.5638	6.5608	6.558	6.5552	6.5526	6.5501	6.5477	6.5455
15	6.6411	6.6369	6.6329	6.6289	6.6252	6.6215	6.618	6.6146	6.6113	6.6082	6.6052	6.6023	6.5996	6.5969	6.5944	6.5921
16	6.6955	6.691	6.6866	6.6824	6.6783	6.6743	6.6705	6.6668	6.6633	6.6598	6.6565	6.6534	6.6503	6.6474	6.6446	6.642
17	6.75	6.7452	6.7405	6.736	6.7316	6.7273	6.7232	6.7192	6.7153	6.7116	6.708	6.7045	6.7012	6.698	6.6949	6.692
18	6.8006	6.7955	6.7907	6.7859	6.7813	6.7768	6.7725	6.7683	6.7642	6.7603	6.7565	6.7528	6.7493	6.7459	6.7426	6.7395
19	6.8511	6.8459	6.8408	6.8358	6.831	6.8263	6.8218	6.8174	6.8131	6.8089	6.8049	6.8011	6.7973	6.7937	6.7903	6.7869
20	6.8876	6.8825	6.8775	6.8727	6.868	6.8634	6.859	6.8547	6.8506	6.8465	6.8427	6.8389	6.8353	6.8318	6.8286	6.8253
21	6.9245	6.9195	6.9147	6.91	6.9054	6.9009	6.8966	6.8925	6.8884	6.8845	6.8808	6.8772	6.8737	6.8703	6.8671	6.864
22	6.9683	6.9633	6.9584	6.9536	6.949	6.9445	6.9402	6.9359	6.9319	6.9279	6.9241	6.9205	6.9169	6.9135	6.9103	6.9072
23	7.0121	7.007	7.0021	6.9973	6.9926	6.988	6.9837	6.9794	6.9753	6.9713	6.9675	6.9638	6.9602	6.9568	6.9535	6.9503
24	7.0436	7.0388	7.0341	7.0296	7.0252	7.0209	7.0168	7.0128	7.0089	7.0052	7.0017	6.9982	6.9949	6.9918	6.9887	6.9858
25	7.0762	7.0716	7.0672	7.063	7.0588	7.0548	7.051	7.0473	7.0437	7.0403	7.0369	7.0338	7.0308	7.0279	7.0251	7.0225
26	7.1078	7.1036	7.0995	7.0956	7.0918	7.0881	7.0846	7.0812	7.078	7.0748	7.0719	7.069	7.0664	7.0638	7.0614	7.0591
27	7.1408	7.137	7.1332	7.1296	7.1261	7.1228	7.1196	7.1166	7.1137	7.1109	7.1083	7.1058	7.1034	7.1012	7.0991	7.0972
28	7.1817	7.1779	7.1743	7.1708	7.1675	7.1643	7.1612	7.1583	7.1555	7.1529	7.1504	7.148	7.1458	7.1437	7.1417	7.1399
29	7.2229	7.2193	7.2158	7.2124	7.2092	7.2061	7.2032	7.2004	7.1978	7.1952	7.1929	7.1906	7.1885	7.1866	7.1847	7.183
30	7.2743	7.2705	7.2668	7.2632	7.2598	7.2565	7.2533	7.2503	7.2475	7.2448	7.2422	7.2397	7.2374	7.2353	7.2333	7.2314
31	7.3257	7.3217	7.3177	7.3139	7.3103	7.3068	7.3035	7.3002	7.2972	7.2942	7.2915	7.2888	7.2863	7.2839	7.2817	7.2796
32	7.3861	7.3816	7.3772	7.3729	7.3688	7.3648	7.3609	7.3572	7.3537	7.3503	7.347	7.3439	7.3409	7.3381	7.3354	7.3328
33	7.4475	7.4424	7.4375	7.4327	7.4281	7.4236	7.4192	7.4151	7.411	7.4071	7.4034	7.3998	7.3963	7.393	7.3898	7.3868
34	7.5024	7.497	7.4917	7.4866	7.4817	7.4769	7.4722	7.4677	7.4633	7.4591	7.4551	7.4512	7.4474	7.4438	7.4403	7.437
35	7.5575	7.5517	7.5462	7.5407	7.5355	7.5303	7.5253	7.5205	7.5158	7.5113	7.5069	7.5027	7.4986	7.4947	7.4909	7.4872

Figure 2: Data Outlook with Headings

4. Feature Engineering

- Feature engineering is a machine learning technique that leverages data to create new variables that aren't in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.
- This way computational time can significantly be cut down, thus improving the performance of the model by transforming the inputs and lowering the number of features.

4.1.Features from Laser Scan Data

- Description of Data
    - The laser range data is a 1D array of 1080 values which is the range recorded at each 0.25deg within a 270deg field of view with the 540th element being the range corresponding to directly in front of the robot.

- Engineering
    - In order to reduce the field of view from 0.25deg/column range to approximately 30deg/column range in front of the robot, select 120 columns (as each column is $0.25°, \therefore 120 \times 0.25° = 30°$) in the center.
    - The above 120 columns are combined by taking their mean. The result produced will be new data with 9 features, which drastically reduces the computational time for training.

4.2.Features from Positional Data

- Description of Data
    - The robot pose and goal information are spatially represented as 3D Cartesian coordinates (x, y, z) for position and a quaternion $(q = qr + qi^2 + qj^2 + qk^2)$ for the orientation.
    - The positional data is arranged as $[x{:}\text{coordinate} + y{:}\text{coordinate} + \text{quaternion}]$ for the final goal, local goal, and current pose.

- Engineering
    - We have disregarded the quaternion's z-coordinate as well as the qi and qj coordinates because the current work only considers the horizontal plane (we only have qr and qk).
    - Therefore, we calculate q by considering qr and qk columns $(q = qr + qk^2)$ and making it a single column.
    - Thus, the number of features is reduced from 12 to 9

4.3.Features to be modeled (Outputs values)

- Commanded Actions (v, ω)
    - The action commands are provided in the form of a translational velocity command (v) and a rotational velocity command (w) which are present in the last two columns of the *.csv data file. The action commands are the output values (Y) in the train and test data sets which the model is being trained upon and whose prediction will be tested.

4.4.Final features

- Description of Data
    - 9 features from the laser scan data
    - 9 features from the positional data
    - 2 output columns that remain unchanged
    - The correctness of the selected features can be cross verified by using a correlation matrix.
    - All the CSV files in the selected folder are combined into one after performing the feature selection procedure mentioned above.

4.5.Feature verification

- Correlation
    - A correlation matrix is simply a table that displays the correlation coefficients for different variables.
    - The column dependencies are revealed by the correlation matrix.
    - If there is no association between any of the blocks in the printed figure, the columns corresponding to those blocks can be eliminated.
    - Code to run this process can be found here link.
    - The image below, however, demonstrates that all of the columns are connected.
    - Additionally, the describe() method can be used to further investigate this correlation. This function gives us some crucial information about the chosen input file.

- Standard deviation
    - Each column's standard deviation is displayed in the ”STD” column.
    - If the standard deviation is zero or extremely near zero, the columns have no impact on the data because they barely change.
    - After our feature selection, all of the columns had standard deviation, therefore we can say that our feature selection was successful.

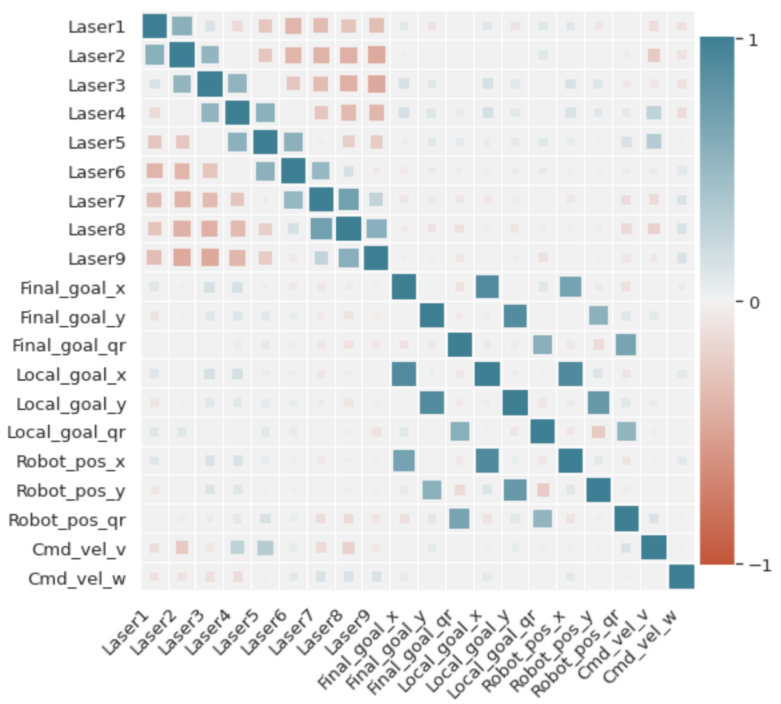
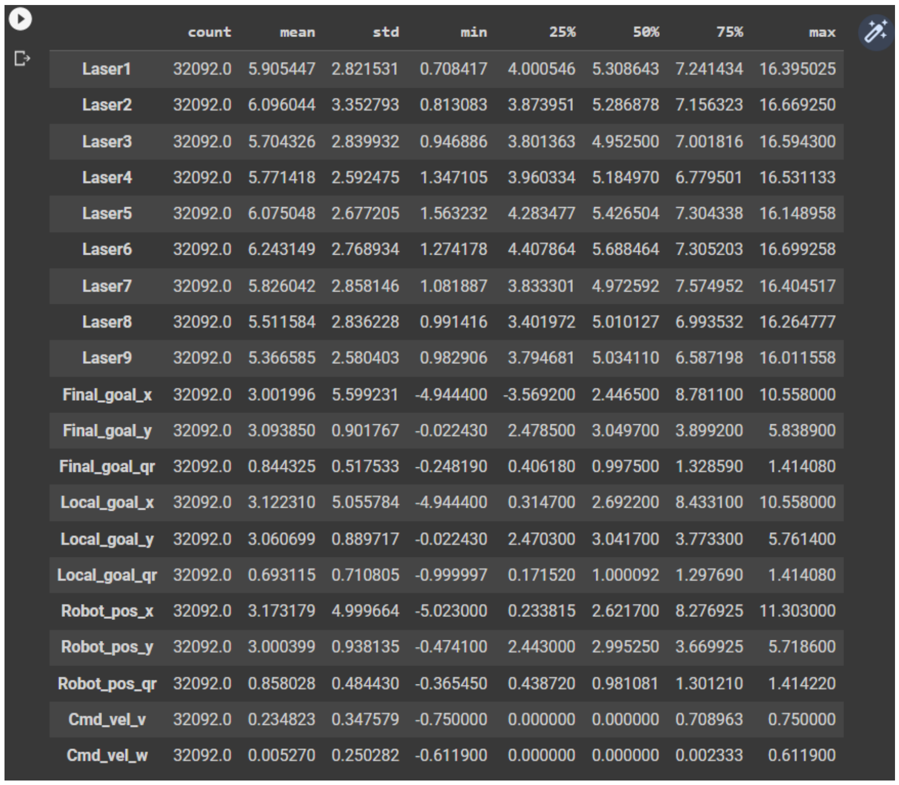


Figure 3: Correlation Heat Map



	count	mean	std	min	25%	50%	75%	max
Laser1	32092.0	5.905447	2.821531	0.708417	4.000546	5.308643	7.241434	16.395025
Laser2	32092.0	6.096044	3.352793	0.813083	3.873951	5.286878	7.156323	16.669250
Laser3	32092.0	5.704326	2.839932	0.946886	3.801363	4.952500	7.001816	16.594300
Laser4	32092.0	5.771418	2.592475	1.347105	3.960334	5.184970	6.779501	16.531133
Laser5	32092.0	6.075048	2.677205	1.563232	4.283477	5.426504	7.304338	16.148958
Laser6	32092.0	6.243149	2.768934	1.274178	4.407864	5.688464	7.305203	16.699258
Laser7	32092.0	5.826042	2.858146	1.081887	3.833301	4.972592	7.574952	16.404517
Laser8	32092.0	5.511584	2.836228	0.991416	3.401972	5.010127	6.993532	16.264777
Laser9	32092.0	5.366585	2.580403	0.982906	3.794681	5.034110	6.587198	16.011558
Final_goal_x	32092.0	3.001996	5.599231	-4.944400	-3.569200	2.446500	8.781100	10.558000
Final_goal_y	32092.0	3.093850	0.901767	-0.022430	2.478500	3.049700	3.899200	5.838900
Final_goal_qr	32092.0	0.844325	0.517533	-0.248190	0.406180	0.997500	1.328590	1.414080
Local_goal_x	32092.0	3.122310	5.055784	-4.944400	0.314700	2.692200	8.433100	10.558000
Local_goal_y	32092.0	3.060699	0.889717	-0.022430	2.470300	3.041700	3.773300	5.761400
Local_goal_qr	32092.0	0.693115	0.710805	-0.999997	0.171520	1.000092	1.297690	1.414080
Robot_pos_x	32092.0	3.173179	4.999664	-5.023000	0.233815	2.621700	8.276925	11.303000
Robot_pos_y	32092.0	3.000399	0.938135	-0.474100	2.443000	2.995250	3.669925	5.718600
Robot_pos_qr	32092.0	0.858028	0.484430	-0.365450	0.438720	0.981081	1.301210	1.414220
Cmd_vel_v	32092.0	0.234823	0.347579	-0.750000	0.000000	0.000000	0.708963	0.750000
Cmd_vel_w	32092.0	0.005270	0.250282	-0.611900	0.000000	0.000000	0.002333	0.611900

Figure 4: Standard Deviation chart

5. Other preprocessing steps

5.1. Data scaling

- ❖ The data given is provided by the sensors used in a real-time application. Hence, 'Scaling' has already been done on the provided data as the developed model will be used in the same environment.
- ❖ So, no further steps such as 'L1 Normalization' or 'Standard Scaling' are required.

5.2. Data Separation

- ❖ The data was priorly separated into training and test datasets at the time of data collection.
- ❖ We will not perform any separations on the data.
- ❖ In this project, we do not intend to perform validation, but only implement training the model and making predictions on the test data.

6. Model Selection and Testing

6.1. Pipeline

The machine learning pipeline that we use to arrive at a final hypothesis that is roughly near to the unidentified target function is shown in the graphic below.

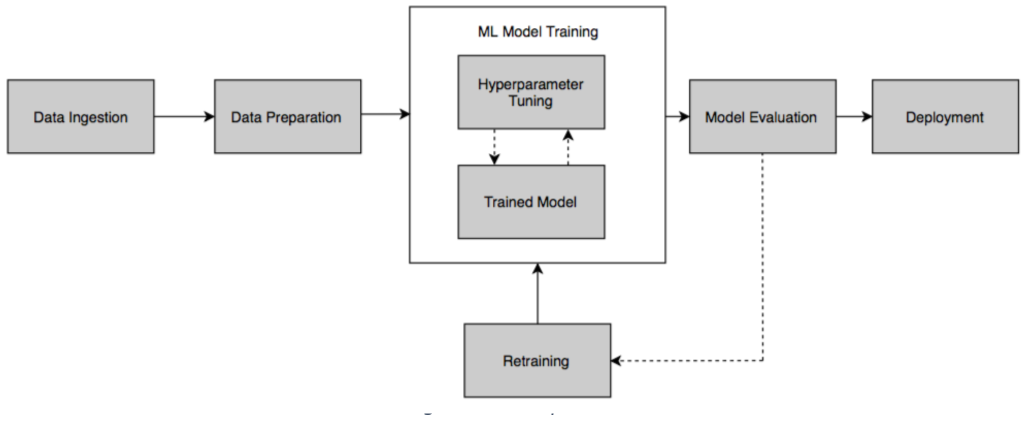


Figure 5: ML Pipeline Flowchart

6.2. Model selection

- As the problem statement calls to anticipate both the linear and angular velocities, the challenge posed here is Regression, as the goal is to predict a continuous value.
- Since multiple regression models are available, they are all evaluated by comparing their performance on the given data set. The model resulting in the least error is finalized.
- For evaluation purposes, only a single data set among the entire collection of CSVs will be taken.
- For the given problem statement, Linear regression, and neural networks are the models used for validation.

6.3.Linear Regression

❖ Description

- Linear regression analysis is used to predict a variable's value based on another variable's value. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.
- Linear regression models are relatively simple and provide an easy-to-interpret mathematical formula that can generate predictions. Linear regression can be applied to various areas in business and academic study.
- Linear regression is defined in terms of a linear function:

$$y' = w_1x_1 + w_2x_2 + \cdots + b$$

where y' is the prediction for the target variable $y, x_1, x_2, \dots, x_k$ are the features $w_1, w_2, \dots w_k$ are the weights and b is a bias term.

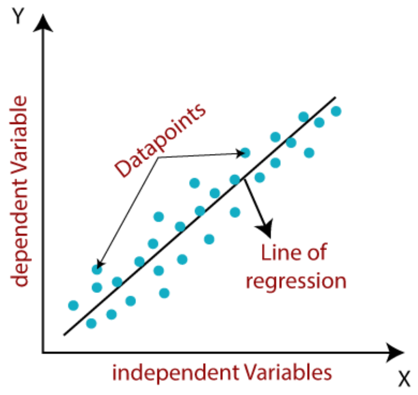


Figure 6: Linear Regression

❖ Code

❖ Results: Error using Linear Regression: We can see from the graph that the E_{in} is lowering and saturating around the value of 0.21560, which is the E_{in} that the linear regression model gives us for the validation data set.

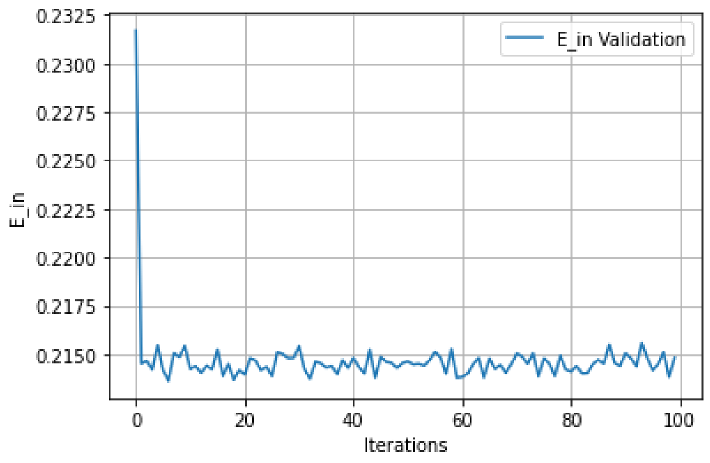


Figure 7: E_in for Linear Regression Model

6.4.Neural Networks (DNN)

❖ Description

- A neural network is a series of algorithms that endeavors to recognize underlying relationships in a data set through a process that mimics how the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.
- **DNN:** Also known as a deep learning network, a deep neural network, at its most basic, is one that involves two or more processing layers. Deep neural networks rely on machine learning networks that continually evolve by compared estimated outcomes to actual results, then modifying future projections.

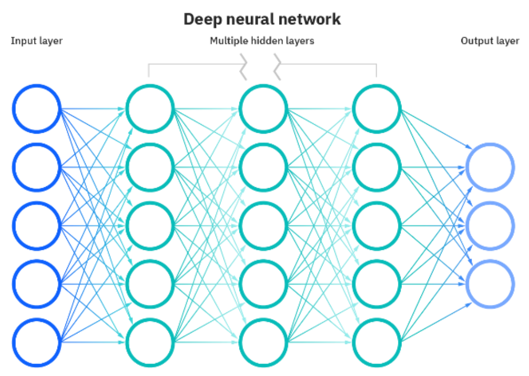


Figure 8: Deep Neural Network Architecture

❖ Code

❖ Results: Upon making predictions on the test data set, we obtain an E_{in} of 0.17345 using the deep neural network model, and the graph shows that the E_{in} is constantly decreasing

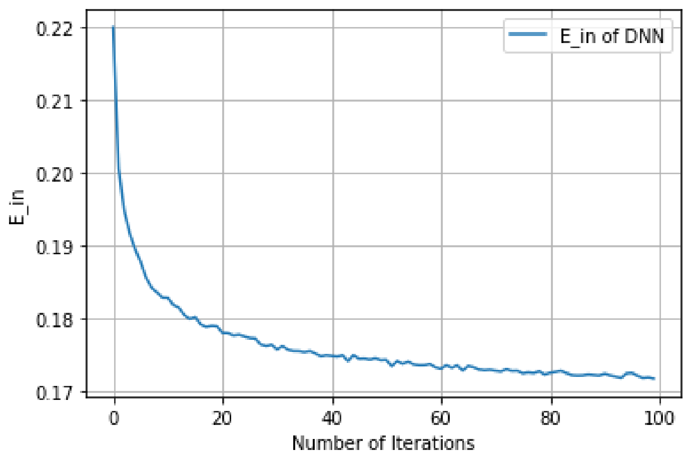


Figure 9: E_in of DNN

6.5.Verification (Final step where we select the model)

We can infer from these findings that the Deep Neural Networking model is superior for solving our problem statement since the evaluated Ein is lower and additional data would enable better model training for a Neural network.

S.no	Model	E_{in}
1	Linear Regression	0.21560
2	DNN	0.17345

7. Training the finalized model

- I have chosen to go with the Deep Neural Network. This Model will be trained on the entire training data, ad it's performance will be calculated based on the prediction the model makes on the test data.
- Code
- E_{in} for Training Data: 0.16977
- E_{in} for Test Data: 0.16876
- Coincidentally, the E_{in} for Training and Test Datasets

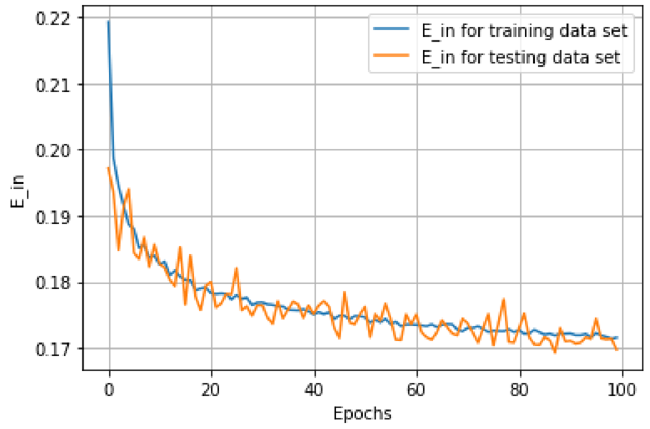


Figure 10: Learning Curves of Finalized model

8. Hyperparameter Tuning

- <u>Hyperparameters</u>: Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The prefix 'hyper' suggests that they are 'top-level' parameters that control the learning process and the model parameters that result from it. These are the parameters that cannot be estimated by the model from the given data. These parameters are used to estimate the model parameters. For example, the learning rate in deep neural networks.

- Tuning: Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within the limits you specify. This process once finished will give you the set of hyperparameter values that are best suited for the model to give optimal results.
- Point to remember: The learning curve is not gradual when we add too many hyperparameters since the model tries to overfit the data. Regularization will help us to solve this
- How to choose the right hyperparameters:
    - It involves trying different combinations of hyperparameters and evaluating their performance on a validation set.
    - This can be done using methods such as grid search or random search, which systematically try different combinations of hyperparameters and evaluate their performance using a metric such as accuracy or mean squared error.
- The Method used here: Tweaking Depth by increasing the number of nodes and layers to improve model fit.
- Since the model does not overfit in this scenario, the hyperparameters chosen can be utilized to train the model.

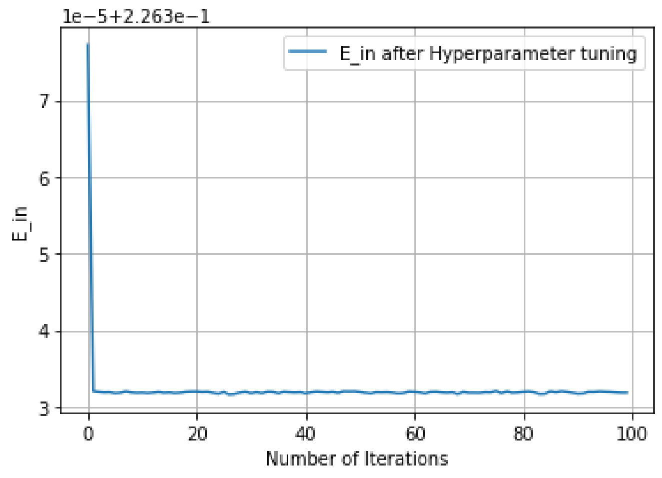


Figure 11: E_in after Hyperparameter Tuning

9. Regularization

- Regularization: It is a technique used in machine learning to prevent over-fitting, which occurs when a model becomes too complex and is not able to generalize well to new, unseen data.
- Procedure to do Regularization:
    - Regularization is typically done by adding a regularization term to the loss function that the model is optimizing.

- The loss function measures how well the model is able to fit the training data, and the regularization term measures the complexity of the model.
- The regularization term is typically controlled by a hyperparameter called the regularization strength, which determines the relative importance of the regularization term compared to the loss term.
- The optimal value of this hyperparameter can be found using methods such as grid search or random search.

❖ Despite the fact that the model does not overfit the data, regularization will make sure that the model is not overfitting the training data set noise.

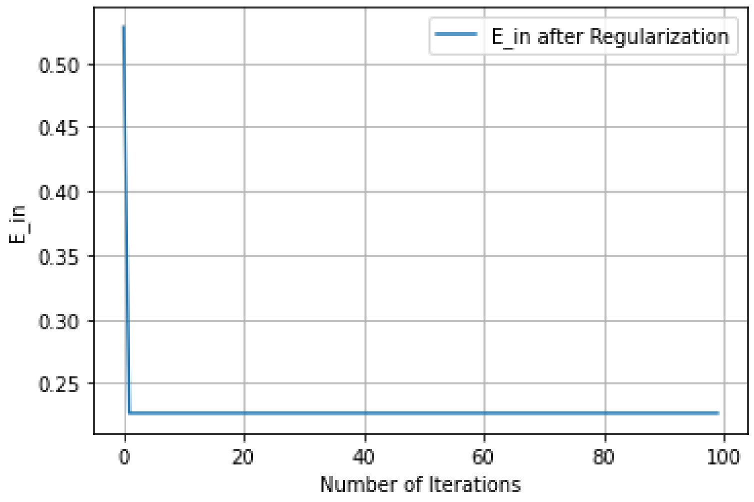


Figure 12: E_in after Regularization

10. Generalization (Estimating E_{out})

❖ Hoeffding's Inequality: Using Hoeffding's inequality we can bound the out-of-sample error, which is the difference between the performance of a model on the training data (in-sample error) and its performance on new, unseen data (out-of-sample error).

❖ The Hoeffding's Inequality can provide a way to select a model that is likely to have good generalization performance on new, unseen data.

❖ E_{out} for the Trained model: 0.24077

❖ As a result, the E_{out} and E_{in} have about the same values. This indicates that the model generalizes well.

11. Challenges faced

- Time was taken to understand how the Machine Learning pipeline works and how various processes contribute to training an optimal model. Lecture notes, slides, and various other online resources greatly aided to develop a proper understanding. Understanding feature extraction.

- Initially, there was a lot of uncertainty with the chosen features. Cross-verification using the correlation matrix provided some confidence.
- As there exist multiple regression models, evaluating each of them posed many challenges in itself. A lot of other models have been worked on apart from the ones mentioned in this report, but due to various coding errors, only the ones with properly organized code have been presented.
- Identifying and tuning hyperparameters also posed a great deal of difficulty, as it is a factor greatly impacts the performance of the system.
- When running the final versions of codes, I ran into a few issues due to inconsistencies. These were successfully identified and sorted.

12. Conclusion

Thus, we have successfully followed the machine learning pipeline to develop a model for predicting the 'command actions' for the car-like robot model. The trained model is computationally reasonable by decreasing the number of features considered and a proper generalization has been shown as indicated by the results.

13. References

[1]. https://www.javatpoint.com/linear-regression-in-machine-learning
[2]. https://www.investopedia.com/terms/n/neuralnetwork.asp
[3]. https://www.ibm.com/cloud/learn/neural-networks
[4]. https://www.ibm.com/topics/linear-regression