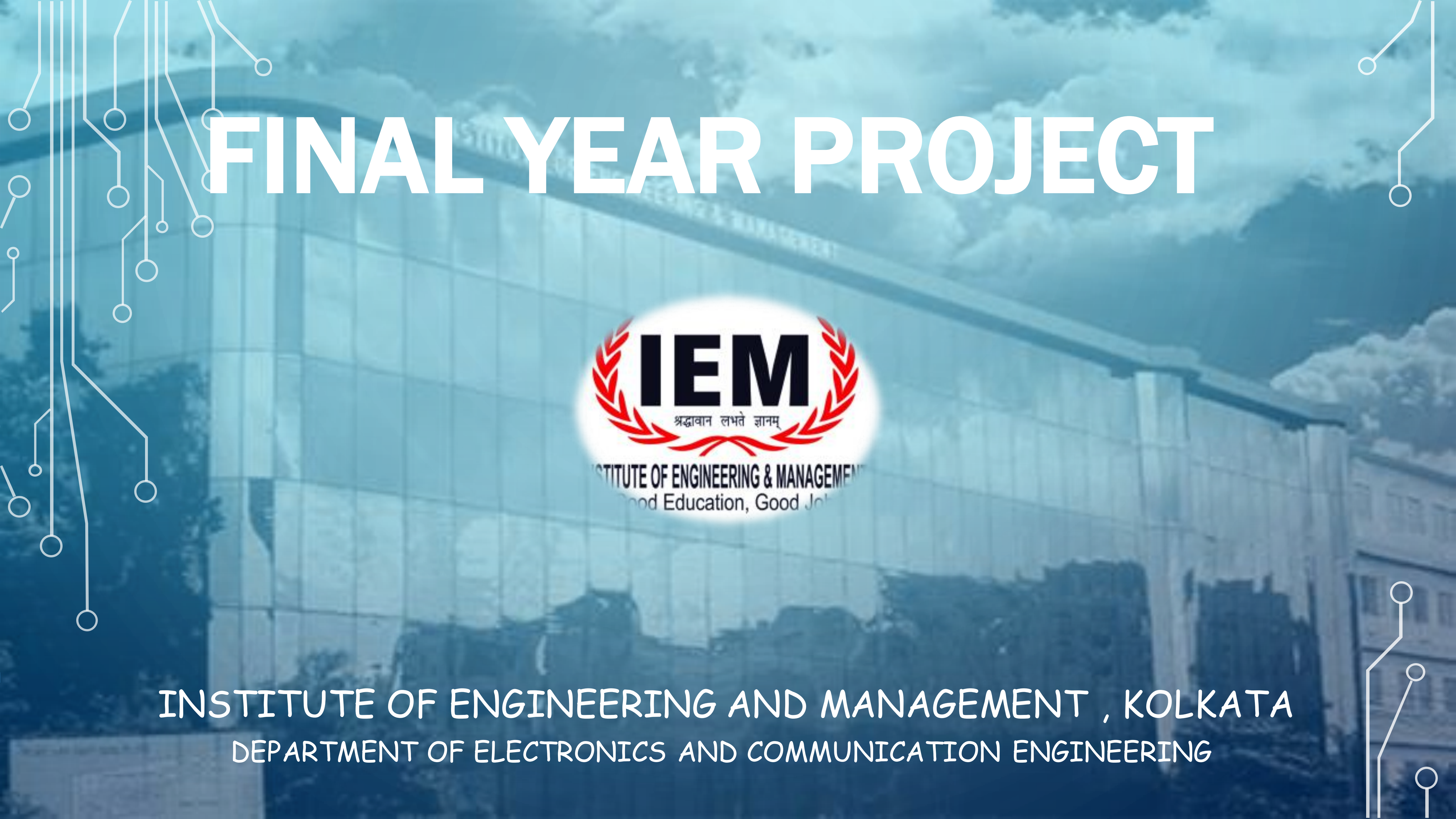


FINAL YEAR PROJECT

INSTITUTE OF ENGINEERING AND MANAGEMENT , KOLKATA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

IEM

TUTE OF ENGINEERING & MANAGEM

Education, Grow

MACHINE LEARNIING MODEL TO AUTOMATE A VEHICLE

Presented By :

1. Akash Bharadwaj

Enrollment No. - 12018002003014
University Roll No. - 10400318193

2. Abhinav Kumar

Enrollment No. - 12018002003018
University Roll No. - 10400318203

3. Ashish Barnwal

Enrollment No. - 12018002003126
University Roll No. - 10400318173

Presented To :

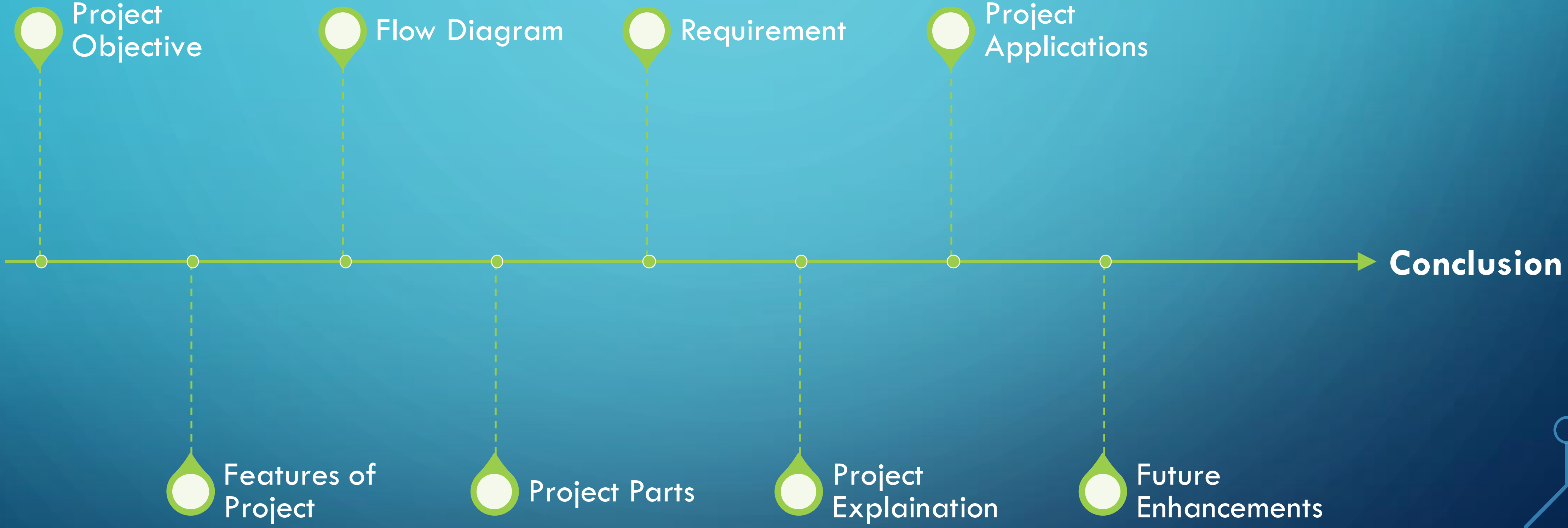
Final Year Project Committee

Department Of Electronic And Communication Engineering
Institute Of Engineering And Management , Kolkata

Internal Advisor :

Prof. Debadyoti Ghosh

PRESENTATION FLOW



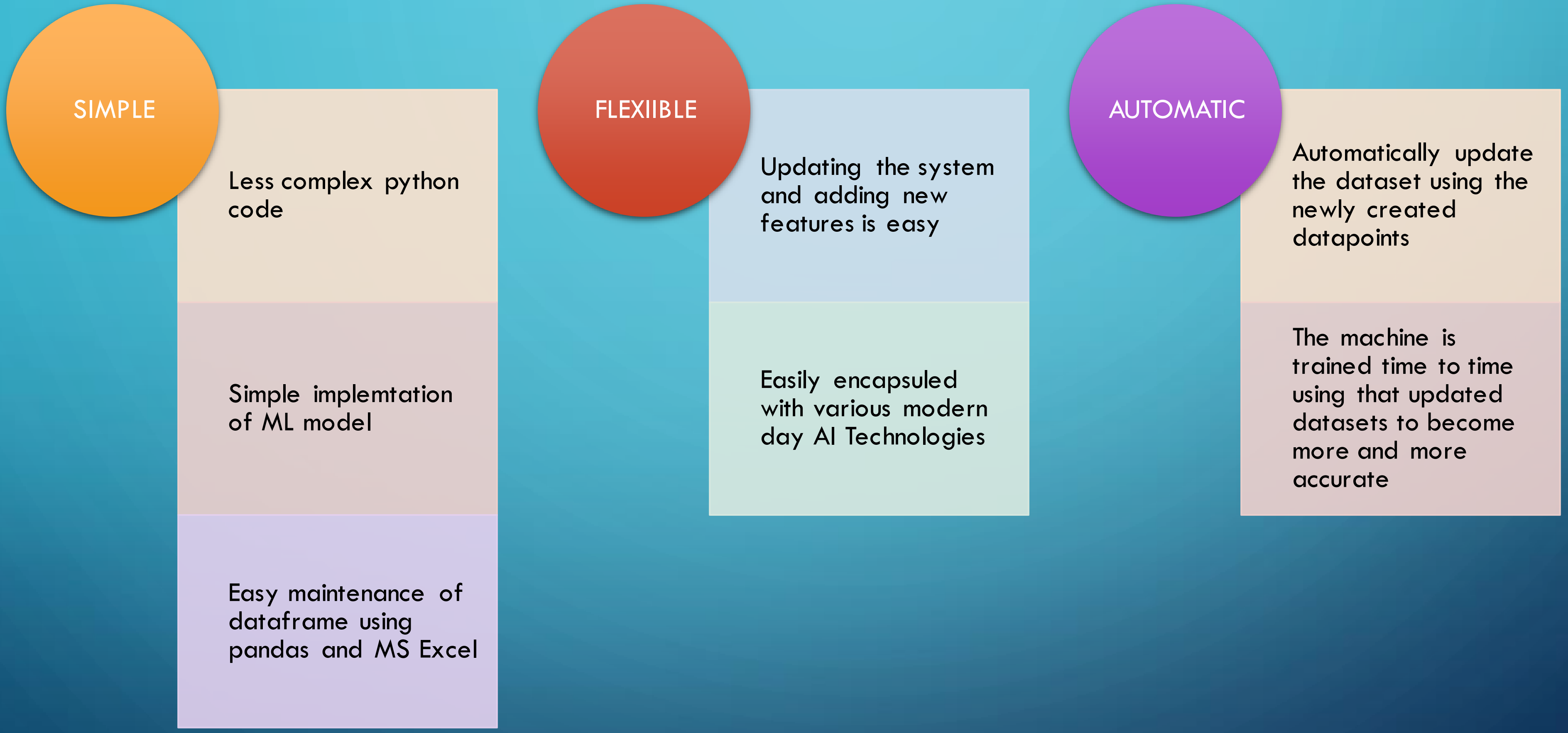
- Project Objective
- Features of Project
- Flow Diagram
- Project Parts
- Requirement
- Project Explaination
- Project Applications
- Future Enhancements
- **Conclusion**

PROJECT OBJECTIVE

To create a decision making system to drive a vehicle by training a machine Learning model using the dataset of decisions taken by an individual on particular conditions .

The dataset is frequently updated by recording the decisions taken by human as well as machine on various scenarios occurred while driving a vehicle.

FEATURES OF PROJECT



SIMPLE

- Less complex python code
- Simple implemtation of ML model
- Easy maintenance of dataframe using pandas and MS Excel

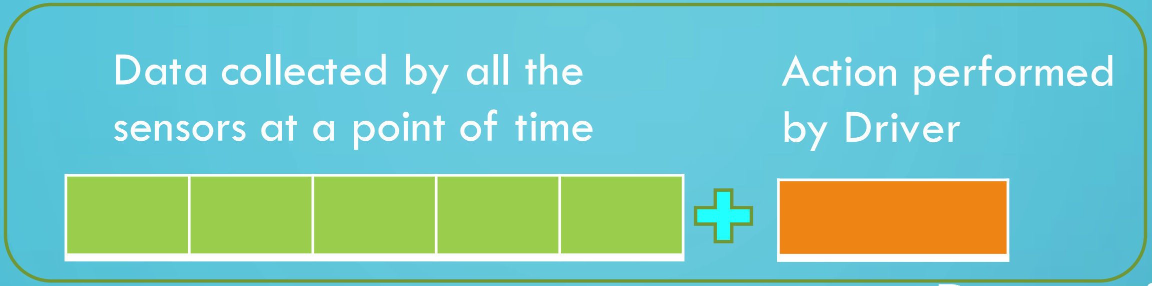
FLEXIIBLE

- Updating the system and adding new features is easy
- Easily encapsuled with various modern day AI Technologies

AUTOMATIC

- Automatically update the dataset using the newly created datapoints
- The machine is trained time to time using that updated datasets to become more and more accurate

FLOW DIAGRAM



Data collected by all the sensors at a point of time

Action performed by Driver

Datapoint

Vehicle **(Not included in this project)**

Sensors for tracking speed , distance , obstacles , motion , etc. are installed in it to note their output and make a datapoint of conditions in front of the Driver

Decision taken by Human

OR

Decision taken by Machine

Removing Duplicate Datapoints

Dataset

Machine Learning Model Training

PROJECT PARTS

Call Volume by Country (sorted)

Visualization Idea 2 - Bar in Bar

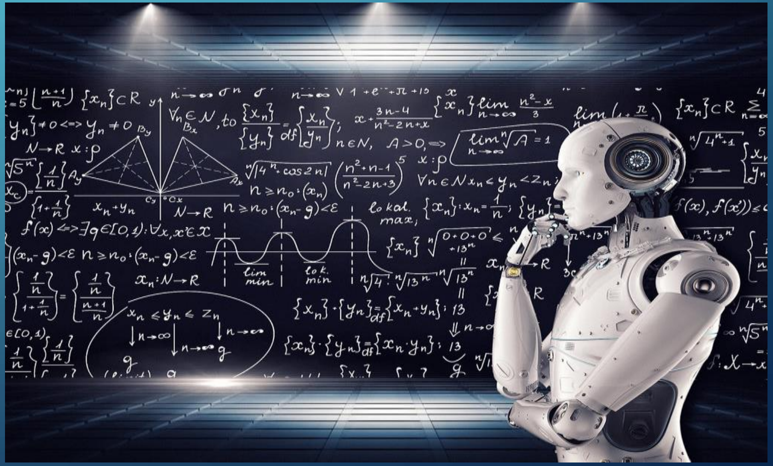
select sort measure	● Total	○ Answered	○ Abandon	○ Aban. in %
IRE	3.681	3.316	365	9,9%
BRA	3.655	3.270	385	10,5%
POR	3.651	3.267	384	10,5%
ES	3.628	3.271	357	9,8%
ARG	3.612	3.217	395	10,9%
FR	3.587	3.224	363	10,1%
NL	3.580	3.190	390	10,9%
UK	3.570	3.186	384	10,8%
AT	3.563	3.218	345	9,7%
BEL	3.536	3.181	355	10,0%
ITA	3.519	3.171	348	9,9%
CHI	3.495	3.123	372	10,6%
MEX	3.480	3.113	367	10,6%
DE	3.443	3.095	348	10,1%
Total	50.000	44.842	5.158	10,3%

❖ *Maintaining core Dataset :*

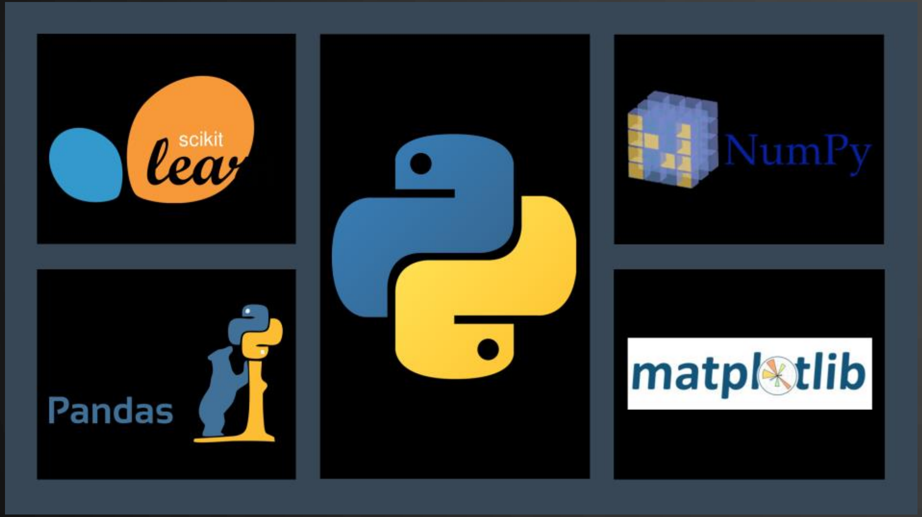
As there is no actual vehicle with sensors to gather those data , so for demonstration of the actual scenario we create datapoint manually and after getting the decision from human or pre-learned ML model on that particular scenario , we just merge that datapoint as one row in our core Dataset.

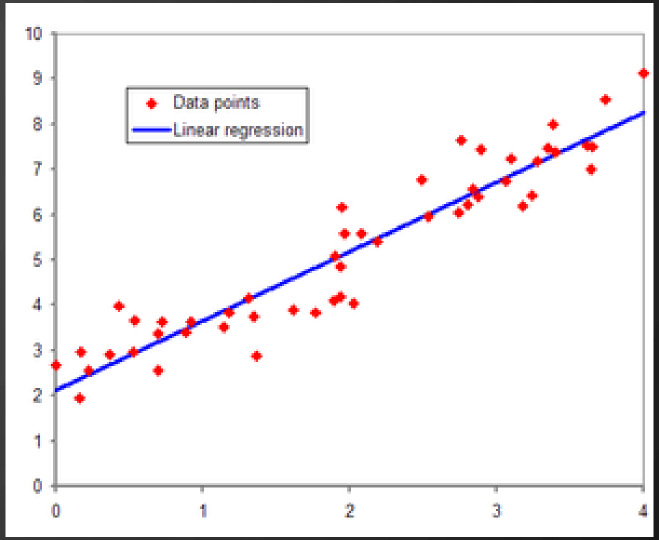
❖ *Training the Machine :*

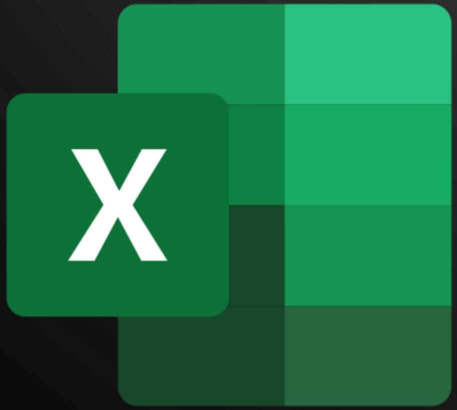
Create a Machine Learning Model based on supervised learning and trained it using core Dataset. Here we import many regression model and choose the model having highest accuracy

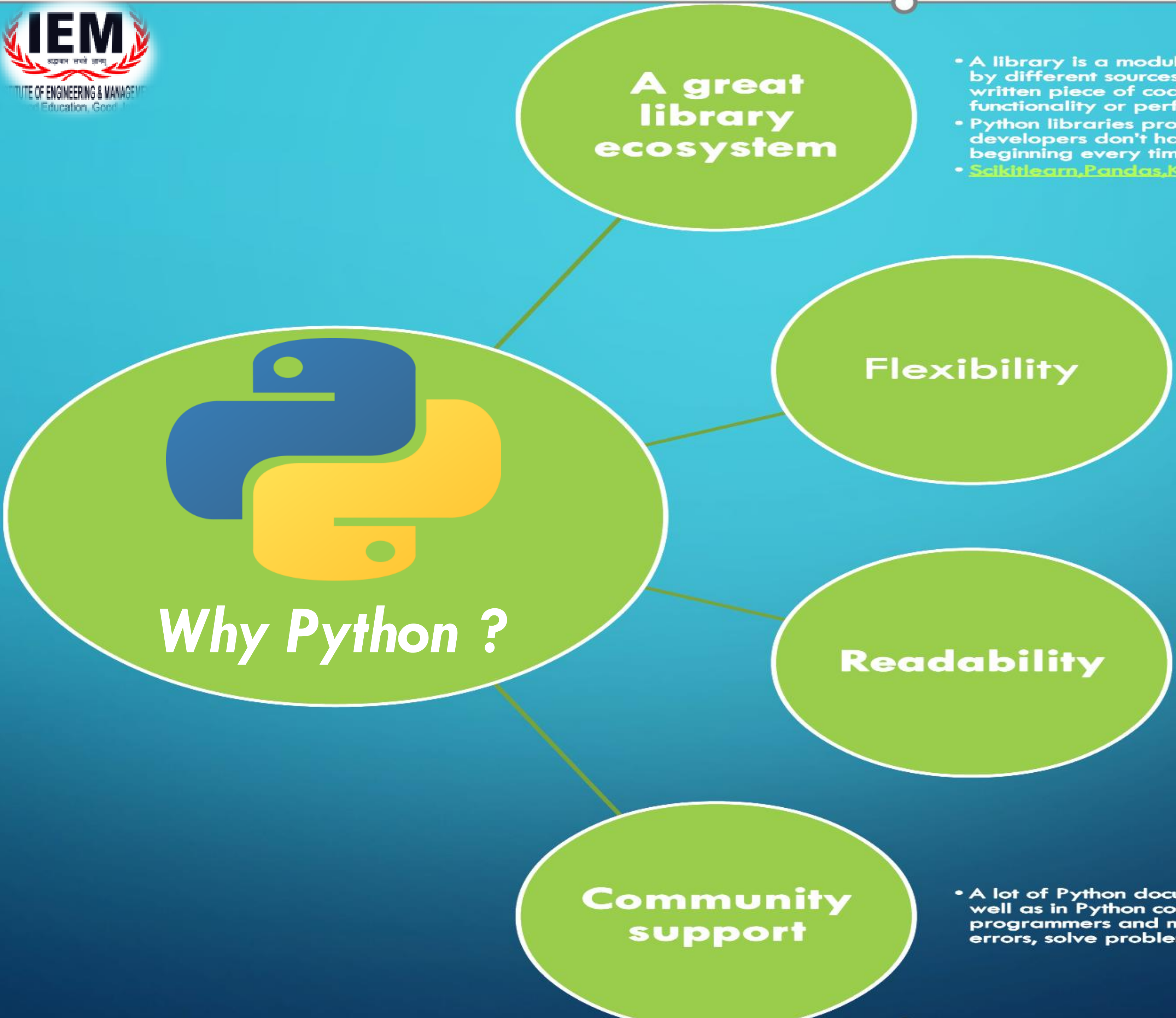


REQUIREMENT









A great library ecosystem

- A library is a module or a group of modules published by different sources like PyPi which include a pre-written piece of code that allows users to reach some functionality or perform different actions.
- Python libraries provide base level items so developers don't have to code them from the very beginning every time.
- Scikitlearn,Pandas,Keras , TensorFlow ,Matplotlib

Flexibility

- It offers an option to choose either to use OOPs or scripting.
- There's also no need to recompile the source code, developers can implement any changes and quickly see the results.
- Programmers can combine Python and other languages to reach their goals.

Readability

- Python programming language resembles the everyday English language, Its simple syntax allows you to comfortably work with complex systems.
- There are also tools like IPython available, which is an interactive shell that provides extra features like testing, debugging, tab-completion, and others, and facilitates the work process.

Community support

- A lot of Python documentation is available online as well as in Python communities and forums, where programmers and machine learning developers discuss errors, solve problems, and help each other out.

Why we used MS Excel ?

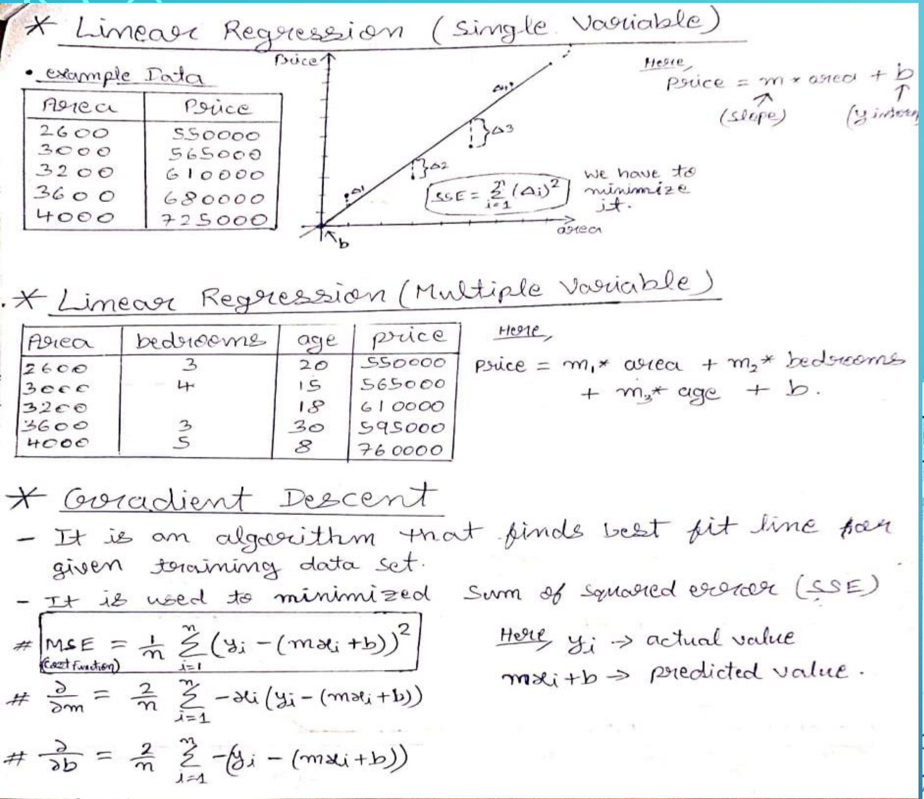
All the dataset managing task , which we can't perform through pandas can be done using Excel.

Storing , Editing and sharing of dataset is easy.

Excel enable easy import and export of core dataset in jupyter Notebook.

MACHINE LEARNING MODEL WE USED IN THIS PROJECT

LINEAR REGRESSION



* Linear Regression (Single Variable)

• example Data

Area	Price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Price
area
b
Δ1
Δ2
Δ3

$\boxed{SSE = \sum_{i=1}^{n} (\Delta i)^2}$ we have to minimize it.

Here, price = m * area + b
(slope) (y interc

* Linear Regression (Multiple Variable)

Area	bedrooms	age	price
2600	3	20	550000
3000	4	15	565000
3200		18	610000
3600	3	30	595000
4000	5	8	760000

Here, price = m_1* area + m_2* bedrooms + m_3* age + b.

* Gradient Descent

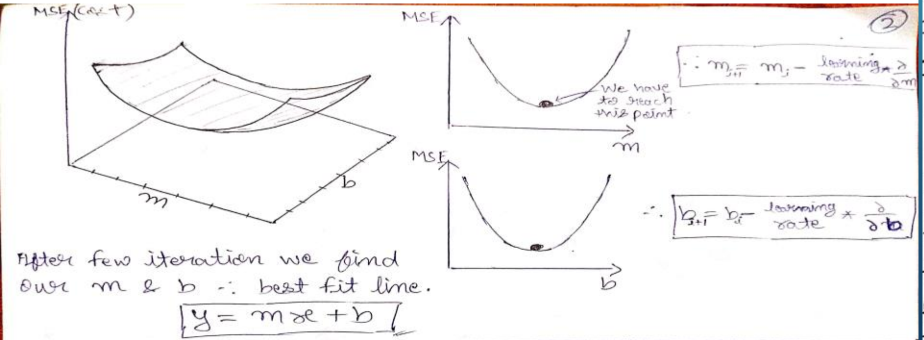
- It is an algorithm that finds best fit line for given training data set.
- It is used to minimized Sum of squared error (SSE)

$\boxed{MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + b))^2}$
(Cost Function)

Here y_i -> actual value
$mx_i + b$ -> predicted value.

$\frac{\partial}{\partial m} = \frac{2}{n} \sum_{i=1}^{n} -x_i (y_i - (mx_i + b))$

$\frac{\partial}{\partial b} = \frac{2}{n} \sum_{i=1}^{n} -(y_i - (mx_i + b))$



$\boxed{\therefore m_{i+1} = m_i - \text{learning rate} * \frac{\partial}{\partial m}}$

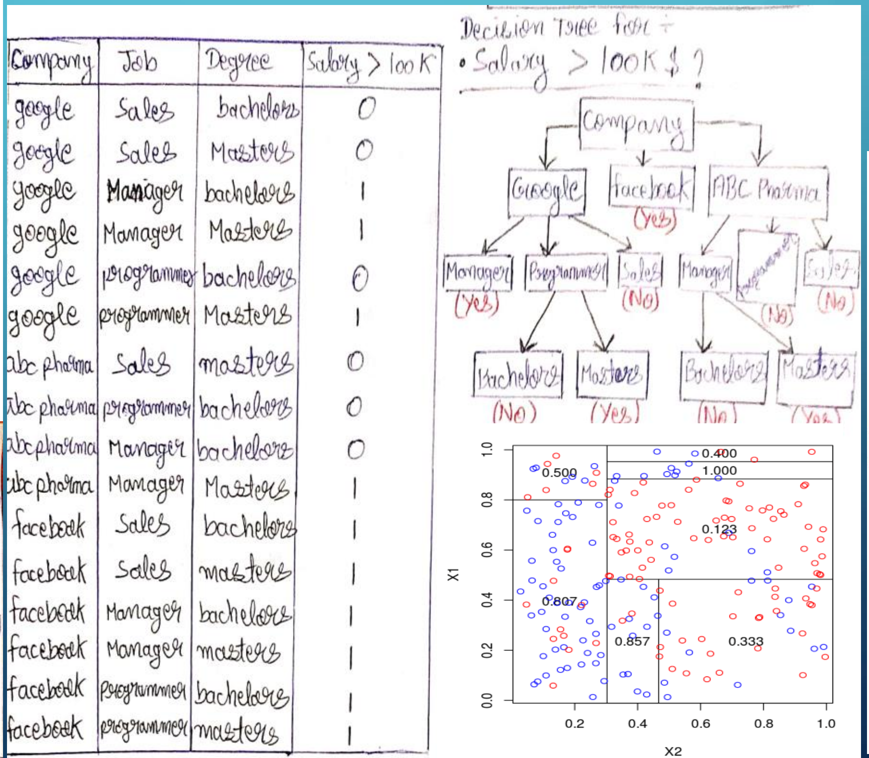
$\therefore \boxed{b_{i+1} = b_i - \text{learning rate} * \frac{\partial}{\partial b}}$

After few iteration we find our m & b ∴ best fit line.

$\boxed{y = mx + b}$

DECISION TREE REGRESSION

This algorithm split datasets again and again to come up with the decision boundaries.

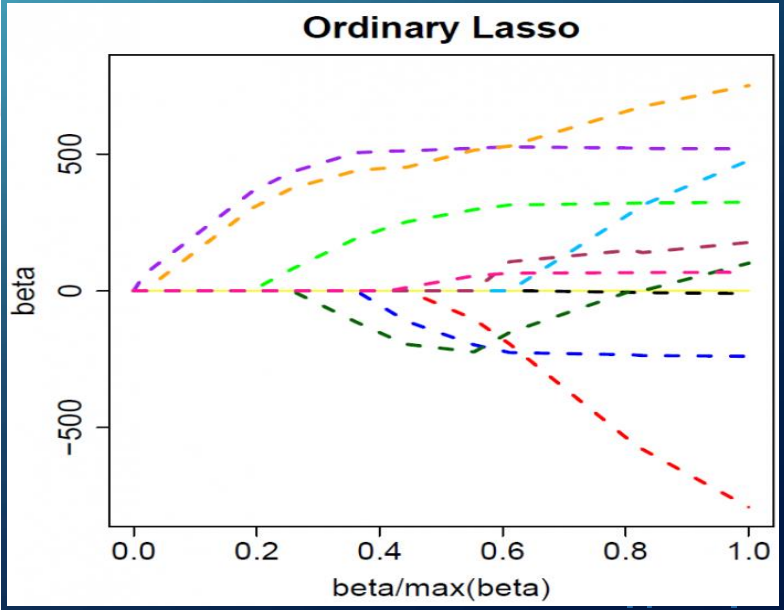


Company	Job	Degree	Salary > 100K
google	Sales	bachelors	0
google	Sales	Masters	0
google	Manager	bachelors	1
google	Manager	Masters	1
google	programmer	bachelors	0
google	programmer	Masters	1
abc pharma	Sales	masters	0
abc pharma	programmer	bachelors	0
abc pharma	Manager	bachelors	0
abc pharma	Manager	Masters	1
facebook	Sales	bachelors	1
facebook	Sales	masters	1
facebook	Manager	bachelors	1
facebook	Manager	masters	1
facebook	programmer	bachelors	1
facebook	programmer	masters	1

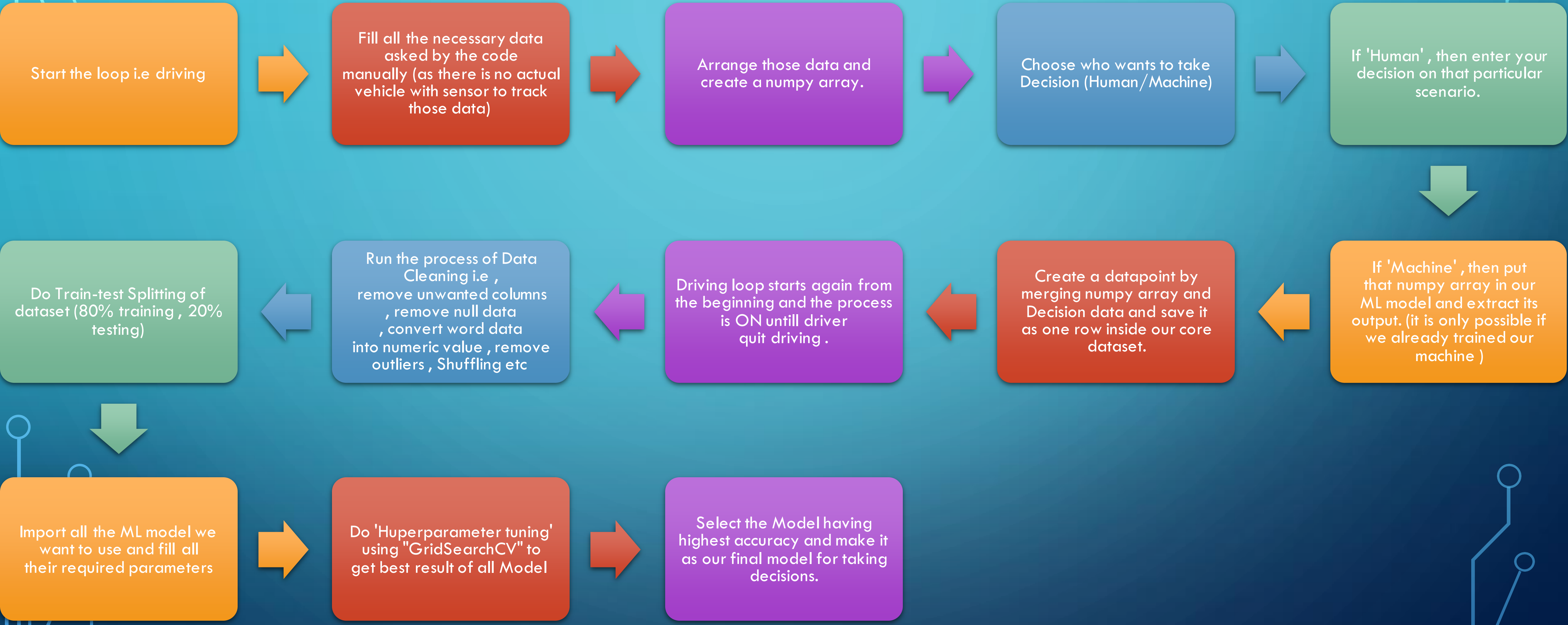
Decision Tree for :
• Salary > 100K $?

LASSO REGRESSION

It is **a type of linear regression that uses shrinkage**. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters)



How the code Works ?



Start the loop i.e driving

Fill all the necessary data asked by the code manually (as there is no actual vehicle with sensor to track those data)

Arrange those data and create a numpy array.

Choose who wants to take Decision (Human/Machine)

If 'Human' , then enter your decision on that particular scenario.

If 'Machine' , then put that numpy array in our ML model and extract its output. (it is only possible if we already trained our machine)

Create a datapoint by merging numpy array and Decision data and save it as one row inside our core dataset.

Driving loop starts again from the beginning and the process is ON untill driver quit driving .

Run the process of Data Cleaning i.e , remove unwanted columns , remove null data , convert word data into numeric value , remove outliers , Shuffling etc

Do Train-test Splitting of dataset (80% training , 20% testing)

Import all the ML model we want to use and fill all their required parameters

Do 'Huperparameter tuning' using "GridSearchCV" to get best result of all Model

Select the Model having highest accuracy and make it as our final model for taking decisions.

Specifications

Components of our Code

- Infinite while Loop (Breaks when condition satisfied)
- If-Else conditional statements
- Input and print statements
- Numpy (np.array([]) , np.append())
- Pandas (df = pd.Datafrrame() , df.append() , df = pd.read_excel() , df.to_excel())

Columns of our Dataset

- 'Driving style'
- 'vehicle side'
- 'Front way is clear upto distance (in meters)'
- 'Vehicle moving alongside'
- 'Vehicle(alongside) speed'
- 'Vehicle(in front) speed(km/hrs)'
- 'Vehicle(in front) side is clear upto distance (in meters) (overtake situation)'
- 'Vehicle(approaching from behind) speed(km/hrs)'
- 'Vehicle(approaching from behind) distance (in meters")'
- 'Response Time (in sec)'
- 'Want to turn'
- 'diversion way available'
- 'Decision Taken by a Driver'

Decisions allowed for the driver to take

- drive straight and maintain same speed
- drive straight and increase speed
- slow down
- Turn right increase speed Turn Left (Overtake)
- Turn left increase speed Turn right (Overtake)
- brake
- emergency brake
- Give pass to the vehicle behind
- Turn Left
- Turn right
- slow down take a side and stop

Scikit-learn components used

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.model_selection import ShuffleSplit
```

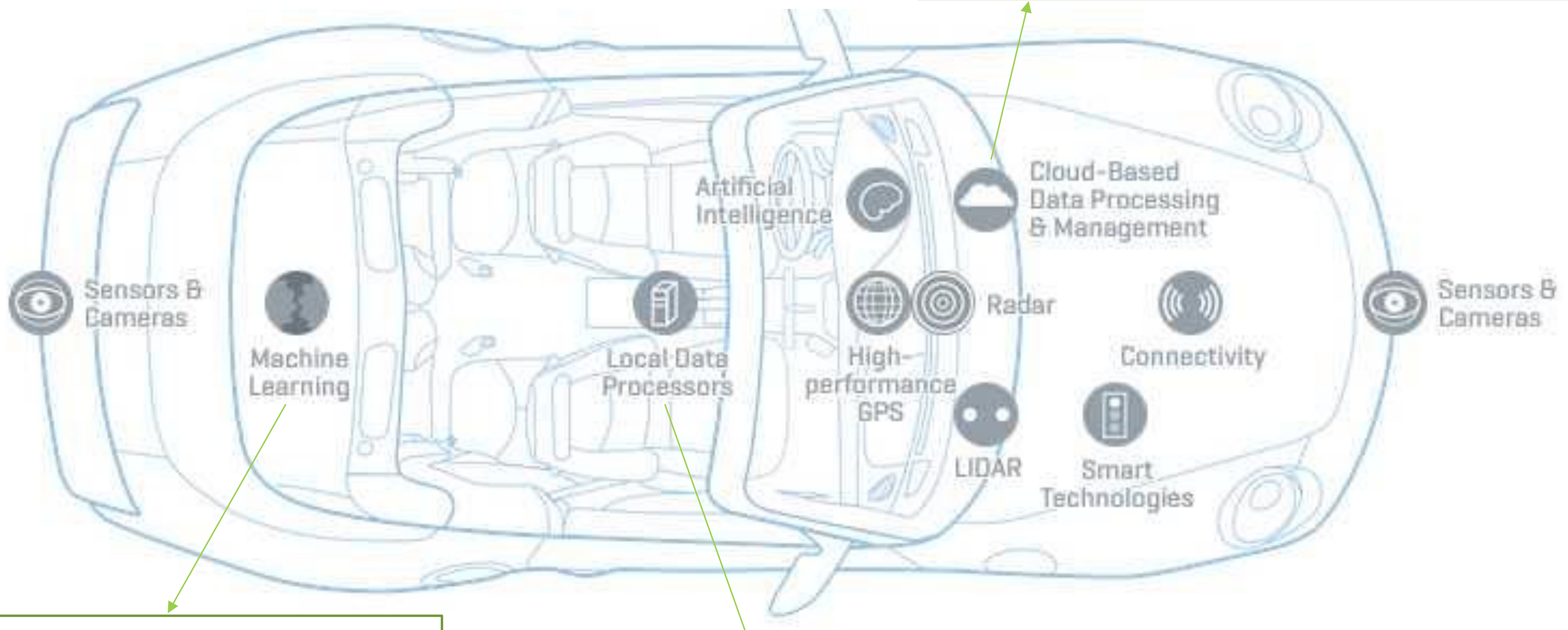
```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.linear_model import Lasso
```

```
from sklearn.tree import DecisionTreeRegressor
```

Project Applications

Our system is also maintaining data of conditions and respective decisions, so that it will further used to train our model again and again for better accuracy

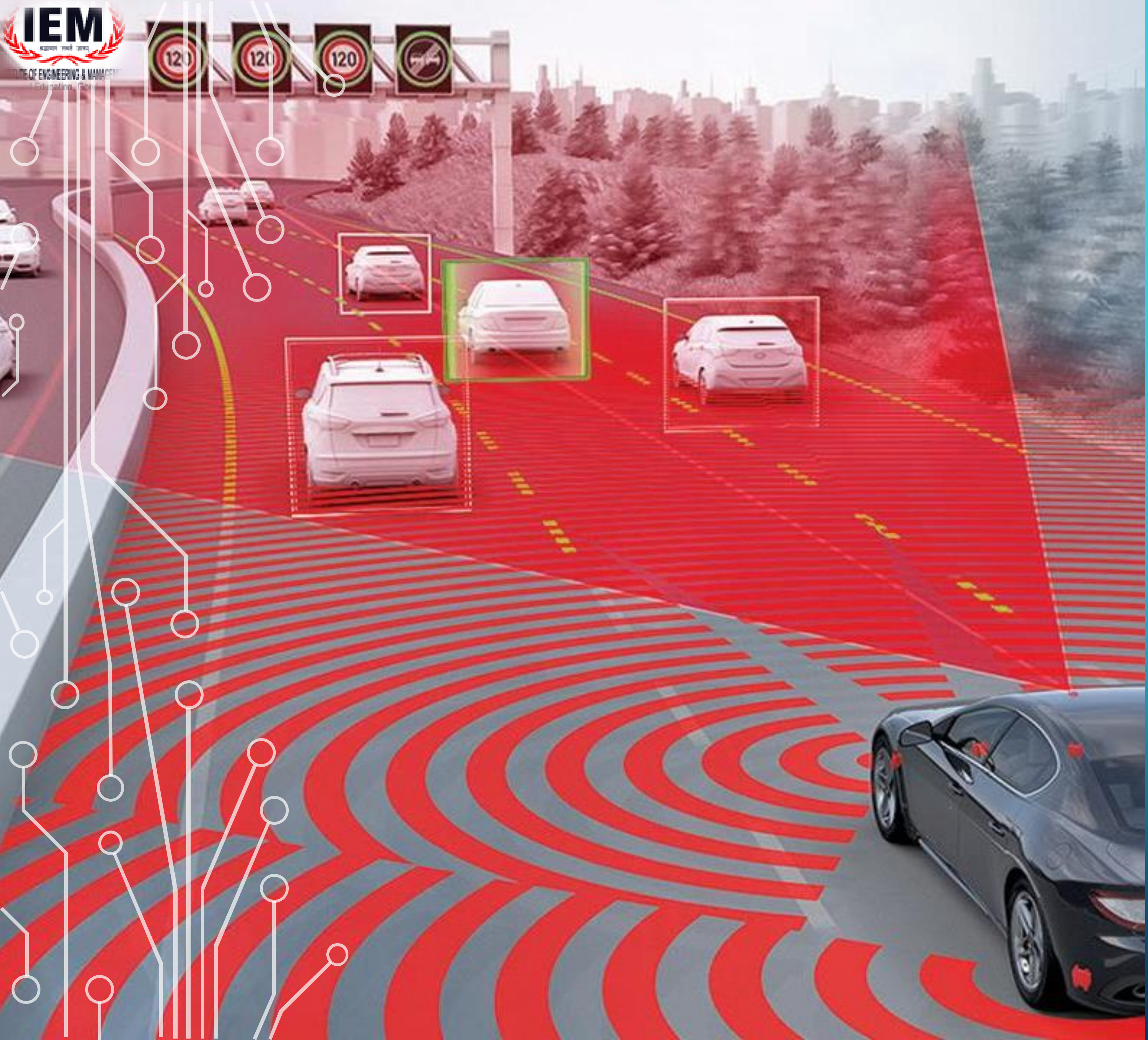


Our system can be easily modified and work as a decision making system which get data about real time scenario from **Local Data Processors** and made decisions according to that.

Our system also collect data from all the sensors , process it and arrange it into an informative manner for better utilization

Limitations

- There is no actual vehicle with sensors to track all those data , so we just assume the scenario and fill all data manually according to that .
- We didn't consider many other parameters .
- If we add any new parameter , we have to do changes manually in the structure of our core dataset as well as in the coding portion also.
- There is no particular interface screen for entering data , everything is happening on jupyter notebook screen.
- There is a space limitation in MS Excel , so using it is not a long term solution for data storing.



FUTURE ENHANCEMENT

- Can Became more Automatic while tracking data.
- We can assume more parameters and track readings according to that.
- We can use this system to automate various kind of robotic vehicle which is cheap as compared to real vehicle.
- We can use a dedicated interface screen rather than jupyter notebook interface.
- We can use a dedicated database to store our core dataset.
- We can use Augmented Reality technology for precise data tracking and accurate decision making.
- We can use many other Regression model .

CONCLUSION

- This project is a simple demonstration of an approach to automate vehicle .
- This is a software based project , Hence there is no Hardware cost required .
- By using this approach we can create a revolutionary AI system which can automate various kind of machine .
- In this project , we try to established a good coordination between MS Excel and Jupyter Notebook , and allow smooth flow of data .
- This project is one of the best example of usage of numpy array , pandas dataframe , Jupyter notebook and various Machine Learning techniques .