

Project Report Format

1. INTRODUCTION

1.1 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

8. TESTING

8.1 Performance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

Source Code

GitHub & Project Demo Link

PROJECT REPORT FORMAT

MACHINE LEARNING BASED VEHICLE PERFORMANCE ANALYZER

TEAM MEMBERS

Gayathri S
Keerthana R
Nandhini S
Sandeep S
Sooraj K.M

1.INTRODUCTION

Automotive Technologies are providing improvised services to the driver's safety and vehicle security under the umbrella of Intelligent Transportation System by using machine learning. In the development of ITS, advanced Automotive Technologies shall play a crucial role in determining the overall experience of users by making it much at ease in terms of reducing the risk of road accidents, risk of cybercrime in the vehicle, buying a used car etc. It is often noted that judging the driver's driving skill is subjective and is difficult to set a standard for driver's skills . The modern approach to transportation system is focusing on rapidly evolving with the intelligent vehicles. High rise in recorded traffic density, road accidents and crisis faced in regulating the effective management of traffic control in urban and rural areas have concerned us to develop a smart solution in context to ITS. The automotive industry has great expectations from these futuristic solutions to improve the safety of people and security of vehicles.

1.1 Purpose

The purpose of this analysis is to help Mechacar's Manufacturing team to understand what car features impact car performance the most. The manufacturing team will incorporate the insights into the manufacturing process aiming to produce the best performing cars in the market, rebrand the company image, and regain market share.

2.LITERATURE SURVEY

2.1 Existing problem

- Normal Wear and Tear

There are some problems that will always happen to any type of car – it doesn't matter about the make or model, the quality of the manufacturing process or how well you maintain the vehicle.

- A Warning Light Shows

Warning lights appear when one of the sensors detect an error and highlights it to the engine control unit.

- The Engine is Sputtering

There are multiple parts that keep an engine running well but a misfiring or sputtering engine is one of the most common issues.

2.2 Reference

Abdullah, M. A.; Jamil, J. F.; Salim, M. A. 2015-12-01

Vehicle dynamic is the effects of movement of a vehicle generated from the acceleration, braking, ride and handling activities. The dynamic behaviours are determined by the forces from tire, gravity and aerodynamic which acting on the vehicle. This paper emphasizes the analysis of vehicle dynamic performance of a real vehicle. Real driving experiment on the vehicle is conducted to determine the effect of vehicle based on roll, pitch, and yaw, longitudinal, lateral and vertical acceleration. The experiment is done using the accelerometer to record the reading of the vehicle dynamic performance when the vehicle is driven on the road. The experiment starts with weighing a car model to get the center of gravity (COG) to place the accelerometer sensor for data acquisition (DAQ). The COG of the vehicle is determined by using the weight of the vehicle. A rural route is set to launch the experiment and the road conditions are determined for the test. The dynamic performance of the vehicle are depends on the road conditions and driving maneuver. The stability of a vehicle can be controlled by the dynamic performance analysis.

2.3 Problem Statement Definition

Vehicle tracking is the process of locating a moving vehicle using a camera. Capture vehicl in

video sequence from surveillance camera is demanding application to improve tracking performance. This technology is increasing the number of applications such as traffic control, traffic monitoring, traffic flow, security etc. The estimated cost using this technology will be very less. Video and image processing has been used for traffic surveillance, analysis and monitoring of traffic conditions in many cities and urban areas. Various methods for speed estimation are proposed in recent years. All approaches attempt to increase accuracy and decrease cost of hardware implementation. The aim is to build an automatic system that can accurately localise and track the speed of any vehicles that appear in aerial video frames.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to 1) create a shared understanding of user needs, and 2) aid in decision making.

Summary

Visualizing user attitudes and behaviors in an empathy map helps UX teams align on a deep understanding of end users. The mapping process also reveals any holes in existing user data.

Format

Traditional empathy maps are split into 4 quadrants (Says, Thinks, Does, and Feels), with the user or persona in the middle. Empathy maps provide a glance into who a user is as a whole and are not chronological or sequential.

3.2 Ideation and Brainstorming

Brainstorming can be used to generate possible solutions for simple problems, but it is unrealistic to expect it to accomplish most problem-solving or planning tasks. The technique is of value as part of a larger effort that includes individual generation of information and ideas and subsequent compilation, evaluation, and selection.

Brainstorming can be used to generate components of a plan, process, solution, or approach and to produce checklists.

THE GROUP

The optimum size for a brainstorming group seems to be six to twelve members, and the optimum group consists of women as well as men. Brainstorming is a total-group effort. Breaking into smaller groups would defeat the purpose of the brainstorming session.

BEGINNING

Prior to the actual session, group members should be provided with a one-page memorandum that states the problem to be considered and outlines the brainstorming procedure.

At the beginning of the actual session, if group members are not already acquainted with one another, they should be introduced (a getting-acquainted activity can be used for this). It is a good idea to conduct a warm-up activity, with the group members directed to brainstorm solutions to a simple problem that is unrelated to the topic of the actual .

THE PROCESS

The leader begins the work session by stating the problem or topic in specific, not general, terms. The problem should be simple rather than complex, so that the group can focus on a single target. The leader should have a list of categories, classifications, or leads (new uses, adaptation, modification, increase, decrease, substitute, rearrange, combine) that can be suggested to the group members if they seem to be getting off track. The leader also can have a few ideas about solutions ready to throw in when the group seems to lag.

It seems to work best if one idea at a time is offered by any one member. This allows all members the space to participate and encourages “piggybacking” on previous ideas.

3.3 Proposed Solution

Performance measures in this chapter are grouped into categories. Transportation can be measured in terms of its primary functions such as safety, accessibility and mobility. It can also be measured in terms of its impact or consequence, such as on the environment, and socioeconomics. It should be noted that while the performance measures identified below are assigned to a single primary category, some measures relate to multiple objectives and categories.

This chapter is limited to performance measures commonly reported out using APM methods and tools. These measures are used in plans and projects to identify needs, compare scenarios and alternatives, and identify benefits and impacts. The chapter focuses on facility level performance measures. System level performance measures generated by APM tools are discussed at a higher level. The performance measures covered are:

- Mobility
- Reliability
- Level of Service (LOS)

- Accessibility
- Safety
- Other Multimodal Performance Measures
- Infrastructure

3.4 Problem Solution Fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.

What is the problem of the customer?

Make assumptions of the problems you think your customer has and then go on to validate those assumptions. In the best case scenario, you can provide measurable insight into how big the customer's problem is.

Is the customer aware of the problem?

The more informed the customer is about their problem, the better. It may even be the case that your customer has realized an inefficient homegrown solution to try to solve the problem. Try as hard as you can to identify how well the customer is aware of the problem and when the customer will run into the problem.

What is your product or service?

Once you have found a validated problem you can realize your solution. Describe what the benefits and functionalities of your product or service are.

What are alternative solutions for the customer?

In a blue ocean, there may be few/no alternative solutions, but in a red ocean, you are bound to have competition trying to solve the same problem. Map out what the competition is and what they offer.

4.REQUIREMENT ANALYSIS

4.1 Functional Requirement

This presents functional requirements for the portion of vehicle performance analyzer that provides crash alerts and advisories to drivers to assist them in avoiding or reducing the harm

associated section with lateral drift and curve-overspeed crashes.

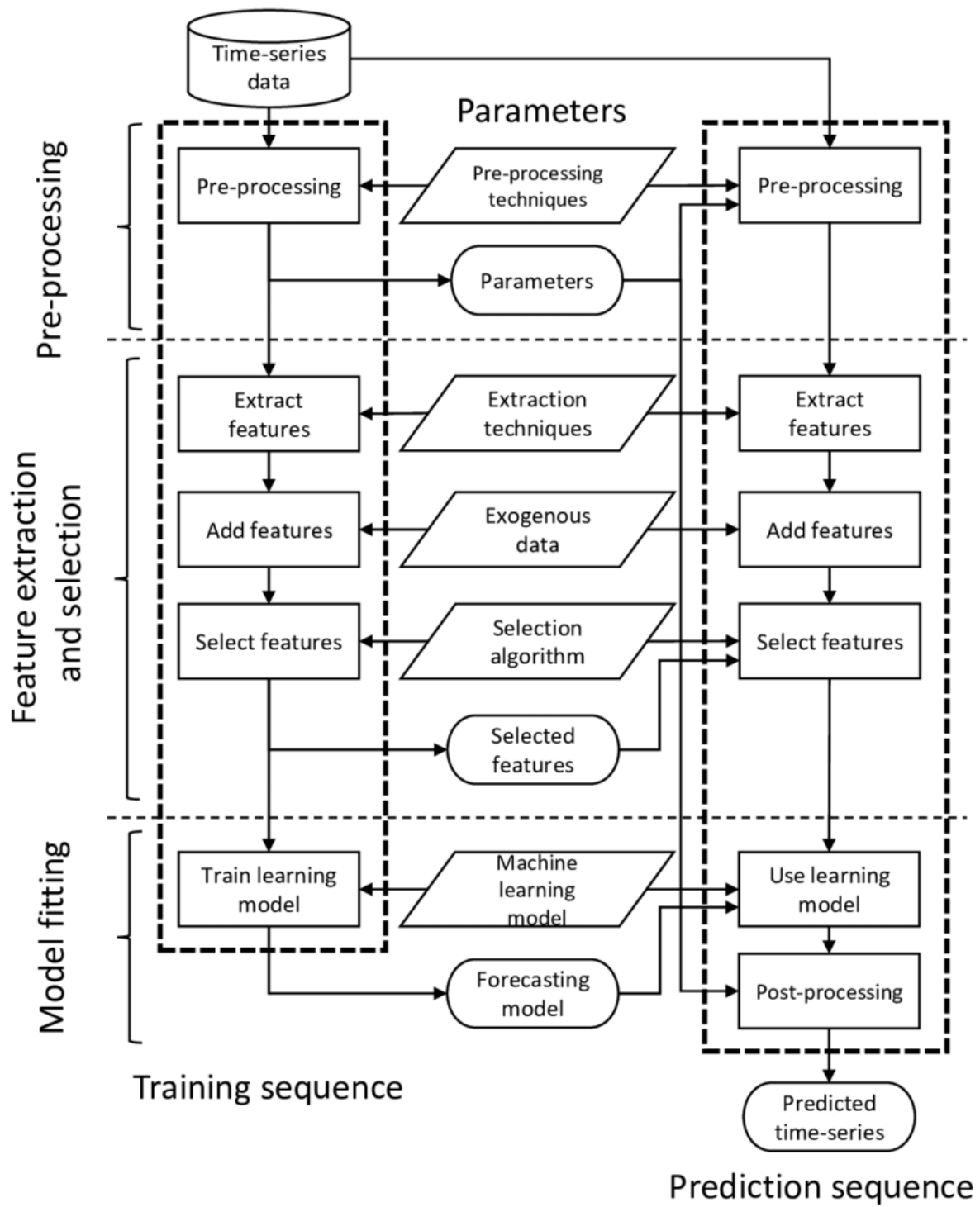
4.2 Non-Functional Requirements

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

5.PROJECT DESIGN

5.1 Data flow Diagrams

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.



5.2 Solution & Technical Architecture

Solution AEs define the Solution Context and collaborate with Solution Management to develop the Solution Vision, Solution Roadmap, and the Capabilities required to meet them. They also work with Solution Management to align the Solution Train's ARTs and Suppliers on what to build and how to build it by establishing the Solution Intent repository. And they play a critical role in solution train events, including Pre- and Post-PI Planning, Solution and System Demos, the Solution Train Sync, and the ART and solution train Inspect and Adapt (I&A) Workshops.

5.3 User Stories

A user story is a structured natural language description of requirements. It follows a compact template that describes the type of user, what they want and (optionally) why (Wautelet et al., 2014). User stories help to capture the description of the software features to be implemented from an end-user perspective. Although many different templates exist, 70% of practitioners use the template (Lucassen et al., 2016a): "As a « type of user », I want « goal », [so that « some reason »]". In our approach, we assume that the considered user stories rely on the following templates:

- "As a « type of user », I want « goal », [so that/so « some reason »]".
- "As a « type of user », I'm able to « goal », [so that/so « some reason »]".

6. PROJECT PANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Estimation is the process for estimating the effort required to complete a prioritized task in the product backlog. This effort is usually measured with respect to the time it will take to complete that task, which, in turn, leads to accurate sprint planning.

- Making teams accountable for deliverables
- Inducing discipline across the Agile team
- Predicting the approximate time it will take to finish a project
- Enabling better sprint management
- Improving team productivity

6.2 Sprint Delivery Schedule

Project Management as a service delivery vehicle plays a pivotal role in executing projects using Agile methodology at Axelerant. We are a team of Project Managers (PMs) who execute big and small projects in staffing and support verticals at Axelerant. We are a close-knit team with our independent views but at the same time ensuring that quality is given paramount importance in whatever activity we do individually and also as a team.

6.3 Reports from Jira

Deliver value to customers faster with real-time insights at your fingertips. Jira Software enables teams to make data-driven decisions with agile reports, dashboards, and more.

7.CODING & SOLUTIONING

HTML

CSS

PYTHON

NUMPY

PANDAS

MATPLOTLIB

7. CODING AND SOLUTIONING

7.1 Feature 1

The first screenshot shows the initial code cell in the notebook. It contains a comment and a line of code to read a CSV file from Google Drive. Below the code, the first five rows of the dataset are displayed as a table.

```
#Reading The Dataset
datas = pd.read_csv("/content/drive/MyDrive/car_performance.csv")
datas.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

The second screenshot shows the next steps in the notebook. It includes a code cell for handling missing values, a variable inspection window, and a code cell for splitting the dataset into training and testing sets.

```
# Handling Missing Values

datas.isnull().any()
```

mpg False
cylinders False
displacement False
horsepower False
weight False
acceleration False
model year False

```
[ ] import numpy as np
import pandas as pd

[ ]

# Splitting The Dataset Into Dependent And Independent Variable.

x = datas.iloc[:,1:8].values

[ ] y = datas.iloc[:,0].values

[ ] # Split The Dataset Into Train Set And Test Set

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
Model_building.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
Connect Editing
[ ] x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
Double-click (or enter) to edit
[ ] # Normalizing
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()
x_train = sd.fit_transform(x_train)
x_test = sd.fit_transform(x_test)
[ ] x_train
array([[ 1.49526939,  1.22961301,  1.24359144, ..., -0.79529768,
        -1.13752513, -0.73301171],
       [-0.85285735, -0.92367663, -1.16092059, ...,  1.24411524,
        -1.41177304,  0.50686898 ],
       [-0.85285735, -0.92367663, -0.68001818, ...,  0.85769009,
        1.85645814,  0.50686898 ],
       ...,
       [-0.85285735, -1.2062235 , -1.45480539, ...,  1.42950823,
        -0.86327722,  0.50686898 ],
       [ 0.32120602,  0.56706235, -0.89224857, ..., -0.2390287 ,
        -1.41177304, -0.73301171],
       [-0.85285735, -0.99180037, -0.86703579, ...,  0.31715028,
        -0.31478141,  0.50686898 ]])
[ ] # Build The Model With The Random Forest Regressor
```

```
Model_building.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
Connect Editing
[ ] # Build The Model With The Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
d = RandomForestRegressor (n_estimators=30,random_state = 0)
d.fit(x_train,y_train)
RandomForestRegressor(n_estimators=30, random_state=0)
[ ] # prediction
y_pred = d.predict(x_test)
y_pred
array([[14.38333333, 24.35666667, 14.21666667, 20.56666667, 18.47333333,
        30.21666667, 34.63333333, 21.15, 16.38333333, 25.76,
        36.60333333, 36.27, 19.53666667, 27.32333333, 16.54333333,
        32.99333333, 28.32333333, 27.49666667, 17.03, 35.82,
        16.47333333, 23.54, 23.16666667, 20.7, 33.69666667,
        26.45, 23.79666667, 30.27333333, 31.93666667, 16.57333333,
        20.26666667, 32.99, 19.79666667, 34.08333333, 20.85666667,
        25.02, 19.65333333, 17.14, 34.78333333, 12.76666667,
        13.73333333, 15.2, 28.32, 32.76666667, 28.74333333,
        22.60666667, 20.54333333, 16.50666667, 23.26, 29.80333333,
        34.31666667, 26.5, 17.63, 27.78333333, 15.96666667,
        12.96666667, 18.06666667, 26.91666667, 31.95666667, 15.68,
        20.81, 25.97, 19.84666667, 21.6, 13.46666667,
        15.33333333, 14.2, 18.90333333, 24.72666667, 14.21666667,
        34.87666667, 13.25, 22.96666667, 18.77666667, 23.83333333,
        32.16666667, 28.17666667, 31.23666667, 31.94, 14.35 ]])
```

```
Model_building.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

y_pred = d.predict(x_test)
y_pred

array([[14.38333333, 24.25666667, 14.21666667, 20.56666667, 18.47333333,
        30.21666667, 34.63333333, 21.15, 16.30333333, 25.76,
        36.68333333, 36.27, 19.53666667, 27.32333333, 16.54333333,
        32.99333333, 28.32333333, 27.49666667, 17.83, 35.82,
        16.47333333, 23.54, 23.16666667, 20.7, 33.69666667,
        26.45, 33.79666667, 30.37333333, 31.93666667, 16.57333333,
        20.26666667, 32.99, 19.79666667, 34.88333333, 20.85666667,
        25.02, 19.65333333, 17.14, 34.78333333, 12.76666667,
        13.73333333, 15.2, 28.32, 32.76666667, 28.74333333,
        22.68666667, 20.54333333, 16.50666667, 23.38, 29.88333333,
        34.31666667, 26.5, 17.63, 27.78333333, 15.96666667,
        12.96666667, 18.06666667, 26.91666667, 31.95666667, 15.68,
        20.81, 25.97, 19.84666667, 21.6, 13.46666667,
        15.33333333, 14.2, 18.90333333, 24.72666667, 14.21666667,
        34.87666667, 13.25, 22.96666667, 18.77666667, 23.83333333,
        32.16666667, 28.17666667, 31.23666667, 31.94, 14.35]])

[ ] from sklearn.metrics import r2_score
accuracy = r2_score(y_test,y_pred)
accuracy

0.8914224871232417

[ ] # save the model
```

```
Model_building.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

[ ] 32.99333333, 28.32333333, 27.49666667, 17.83, 35.82,
16.47333333, 23.54, 23.16666667, 20.7, 33.69666667,
26.45, 33.79666667, 30.37333333, 31.93666667, 16.57333333,
20.26666667, 32.99, 19.79666667, 34.88333333, 20.85666667,
25.02, 19.65333333, 17.14, 34.78333333, 12.76666667,
13.73333333, 15.2, 28.32, 32.76666667, 28.74333333,
22.68666667, 20.54333333, 16.50666667, 23.38, 29.88333333,
34.31666667, 26.5, 17.63, 27.78333333, 15.96666667,
12.96666667, 18.06666667, 26.91666667, 31.95666667, 15.68,
20.81, 25.97, 19.84666667, 21.6, 13.46666667,
15.33333333, 14.2, 18.90333333, 24.72666667, 14.21666667,
34.87666667, 13.25, 22.96666667, 18.77666667, 23.83333333,
32.16666667, 28.17666667, 31.23666667, 31.94, 14.35]])

[ ] from sklearn.metrics import r2_score
accuracy = r2_score(y_test,y_pred)
accuracy

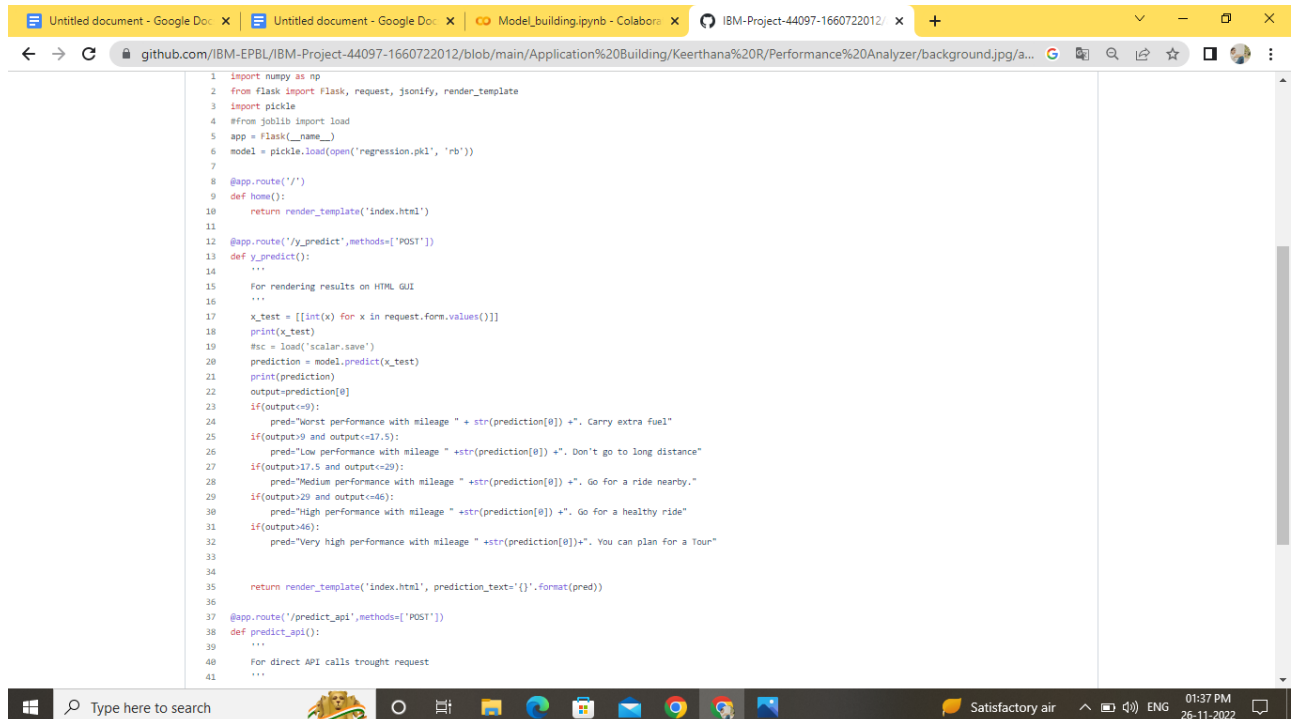
0.8914224871232417

# save the model

import pickle
pickle.dump(d,open('regression.pkl','wb'))

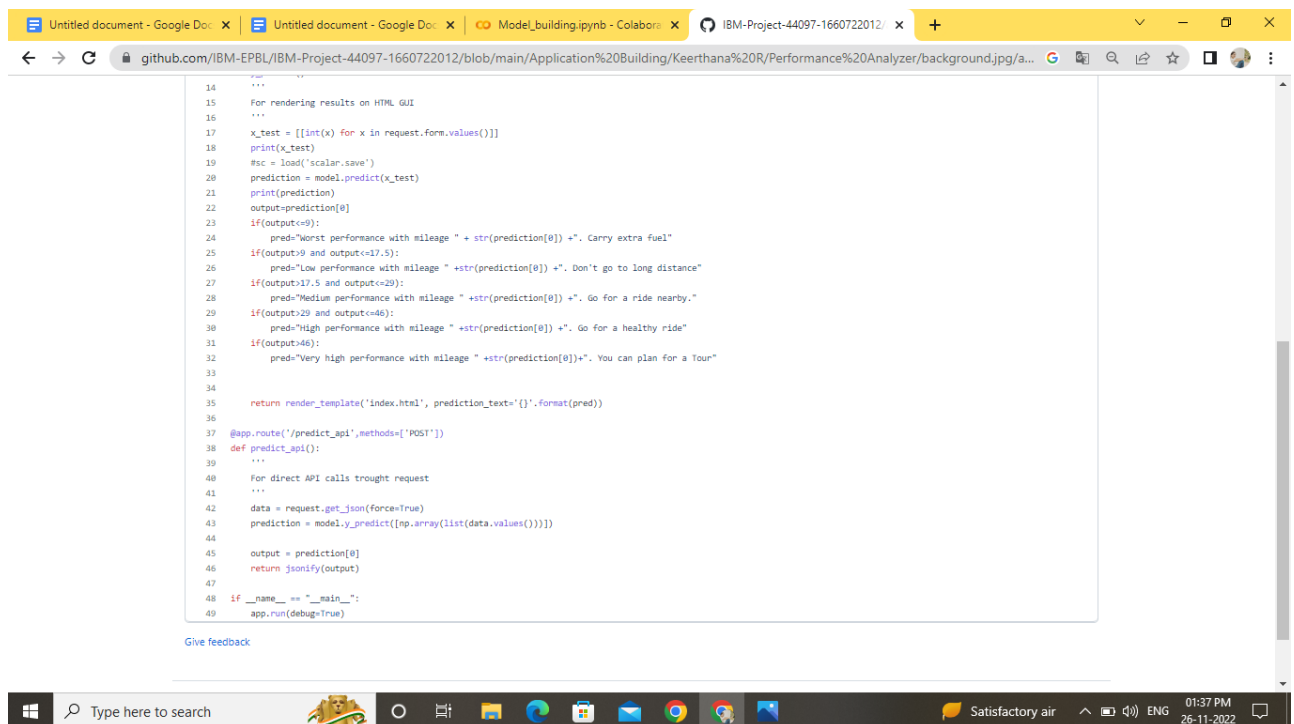
[ ] from google.colab import drive
drive.mount('/content/drive')
```

7.2 Feature 2



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser's address bar displays the URL: `github.com/IBM-EPBL/IBM-Project-44097-1660722012/blob/main/Application%20Building/Keerthana%20R/Performance%20Analyzer/background.jpg/a...`. The notebook contains the following Python code:

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4 #from joblib import load
5 app = Flask(__name__)
6 model = pickle.load(open('regression.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/y_predict', methods=['POST'])
13 def y_predict():
14     """
15     For rendering results on HTML GUI
16     """
17     x_test = [[int(x) for x in request.form.values()]]
18     print(x_test)
19     #sc = load('scalar.save')
20     prediction = model.predict(x_test)
21     print(prediction)
22     output=prediction[0]
23     if(output<9):
24         pred="worst performance with mileage " + str(prediction[0]) + ". Carry extra fuel"
25     if(output>9 and output<17.5):
26         pred="Low performance with mileage " +str(prediction[0]) + ". Don't go to long distance"
27     if(output>17.5 and output<29):
28         pred="Medium performance with mileage " +str(prediction[0]) + ". Go for a ride nearby."
29     if(output>29 and output<46):
30         pred="High performance with mileage " +str(prediction[0]) + ". Go for a healthy ride"
31     if(output>46):
32         pred="Very high performance with mileage " +str(prediction[0]) + ". You can plan for a Tour"
33
34     return render_template('index.html', prediction_text='{}'.format(pred))
35
36 @app.route('/predict_api', methods=['POST'])
37 def predict_api():
38     """
39     For direct API calls through request
40     """
41     data = request.get_json(force=True)
42     prediction = model.y_predict([np.array(list(data.values()))])
43     output = prediction[0]
44     return jsonify(output)
45
46 if __name__ == "__main__":
47     app.run(debug=True)
```



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser's address bar displays the URL: `github.com/IBM-EPBL/IBM-Project-44097-1660722012/blob/main/Application%20Building/Keerthana%20R/Performance%20Analyzer/background.jpg/a...`. The notebook contains the following Python code:

```
14 """
15 For rendering results on HTML GUI
16 """
17 x_test = [[int(x) for x in request.form.values()]]
18 print(x_test)
19 #sc = load('scalar.save')
20 prediction = model.predict(x_test)
21 print(prediction)
22 output=prediction[0]
23 if(output<9):
24     pred="worst performance with mileage " + str(prediction[0]) + ". Carry extra fuel"
25 if(output>9 and output<17.5):
26     pred="Low performance with mileage " +str(prediction[0]) + ". Don't go to long distance"
27 if(output>17.5 and output<29):
28     pred="Medium performance with mileage " +str(prediction[0]) + ". Go for a ride nearby."
29 if(output>29 and output<46):
30     pred="High performance with mileage " +str(prediction[0]) + ". Go for a healthy ride"
31 if(output>46):
32     pred="Very high performance with mileage " +str(prediction[0]) + ". You can plan for a Tour"
33
34     return render_template('index.html', prediction_text='{}'.format(pred))
35
36 @app.route('/predict_api', methods=['POST'])
37 def predict_api():
38     """
39     For direct API calls through request
40     """
41     data = request.get_json(force=True)
42     prediction = model.y_predict([np.array(list(data.values()))])
43     output = prediction[0]
44     return jsonify(output)
45
46 if __name__ == "__main__":
47     app.run(debug=True)
```

Solution for the coding

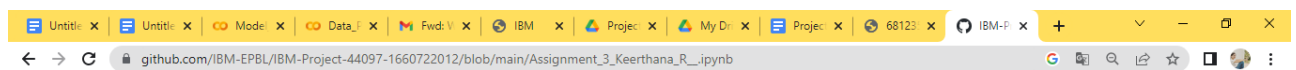


ANALYZE YOUR CAR'S PERFORMANCE



No. of cylinders (count)	Displacement (in miles)	Horsepower (per sec)	Weight (in pounds)	Model Year (YY)	Origin
<input type="button" value="Predict"/>					

{{prediction_text}}



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Building a regression model

1. Download the dataset: Dataset 2. Load the dataset into the tool.

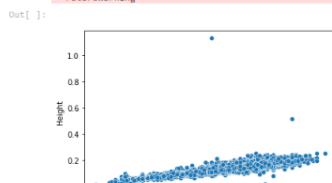
```
In [ ]: df = pd.read_csv('../content/abalone.csv')
df.head()
```

```
Out[ ]:   Sex  Length  Diameter  Height  Whole weight  Shucked weight  Viscera weight  Shell weight  Rings
0  M    0.455    0.365    0.095    0.5140        0.2245        0.1010        0.150        15
1  M    0.350    0.265    0.090    0.2255        0.0995        0.0485        0.070        7
2  F    0.530    0.420    0.135    0.6770        0.2565        0.1415        0.210        9
3  M    0.440    0.365    0.125    0.5160        0.2155        0.1140        0.155        10
4  I    0.330    0.255    0.080    0.2050        0.0895        0.0395        0.055        7
```

3. Perform Below Visualizations. - Univariate Analysis - Bi-Variate Analysis - Multi-Variate Analysis

```
In [ ]: sns.scatterplot(df.Diameter, df.Height)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be "data", and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning
```




```
1 # Importing the required libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import warnings
7 warnings.filterwarnings('ignore')
8
9 # Loading the dataset
10 df = pd.read_csv('IBM-EPBL/IBM-Project-44097-1660722012/blob/main/Assignment_3_Keerthana_R_.ipynb')
11
12 # Checking the data types of the columns
13 df.dtypes
14
15 # Converting the 'date' column to datetime format
16 df['date'] = pd.to_datetime(df['date'])
17
18 # Grouping the data by 'date' and calculating the mean of 'sales'
19 df.groupby('date')['sales'].mean()
20
21 # Plotting the mean sales over time
22 df.groupby('date')['sales'].mean().plot()
23
24 # Checking the distribution of 'sales'
25 df['sales'].hist()
26
27 # Checking the correlation between 'date' and 'sales'
28 df['date'].corr(df['sales'])
29
30 # Checking the correlation between 'date' and 'sales' using Pearson's correlation coefficient
31 df['date'].corr(df['sales'], method='pearson')
```

[illegible]




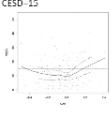
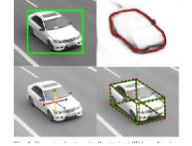
8.TESTING

8.1 Performance Testing

Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMD43198
Project Name	Project - Machine Learning based Vehicle Performance Analyzer
Maximum Marks	10 Marks

Model Performance Testing:
Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	~2	
2.	Accuracy	Training Accuracy - 3 Validation Accuracy 3 	
3.	Confidence Score (Only Yolo Projects)	Class Detected - 4 Confidence Score - CED-12 	

9.RESULTS

9.1 Performance Metrics

Performance metrics are a part of every machine learning pipeline. They tell you if you're making progress, and put a number on it. All machine learning models, whether it's linear regression, or a SOTA technique like BERT, need a metric to judge performance.

10.ADVANTAGES

Machine learning being part of artificial intelligence technology means that the device can learn on its own by analyzing patterns and trends. The IT systems implemented on such machinery, therefore, find solutions to problems by recognizing patterns in databases. Having an AI machine learning in place will then reduce the need for human interaction to run the operations and do the calibrations manually. Having the ability to analyze large volumes of data and coming up with a solution makes the technology highly effective with little interruptions, especially in an industry that time is of the essence.

DISADVANTAGES

The development of these technologies needs a lot of funding. Each step of the way requires an investment to facilitate its success. For one, there has to be a team of developers who come up with the algorithms. Then there is the part where you have to train new people to get conversant with the machine learning language and the implementation process. Lastly, you need specially made machines made for the industry. And all that is quite a massive cost overall.

11. CONCLUSION

In recent years, the number of BEVs is increasing gradually, but the problem of inaccurate residual power display has been restricting the promotion and the use of BEVs. The purpose of this study is to solve the problem of “range anxiety” caused by battery performance and other factors by predicting the BEV driving range. Many studies usually take less factors into account when establishing the prediction model of driving range, which may lead to the poor applicability and prediction accuracy of the model. In this study, a prediction model for BEV driving range based on machine learning has been established. The study is innovative in its application of machine learning method, GBDT algorithm, which includes a very large number of feature variables that cannot be considered by conventional regression methods. Moreover, the study is novel in its accuracy and reliability of a prediction model for BEV driving range.

12. FUTURE SCOPE

To conclude, the car of the future, built according to a new model, will be electric, autonomous and connected. It will bring a number of benefits to society: less pollution, more safety, more free time and services.

SOURCE CODE

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

#from joblib import load

app = Flask(__name__)

model = pickle.load(open('regression.pkl', 'rb'))

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])

def y_predict():

    """

    For rendering results on HTML GUI

    """

    x_test = [[int(x) for x in request.form.values()]]

    print(x_test)

    #sc = load('scalar.save')

    prediction = model.predict(x_test)

    print(prediction)

    output=prediction[0]

    if(output<=9):

        pred="Worst performance with mileage " + str(prediction[0]) + ". Carry extra fuel"

    if(output>9 and output<=17.5):
```

```

pred="Low performance with mileage " +str(prediction[0]) +". Don't go to long distance"

if(output>17.5 and output<=29):

    pred="Medium performance with mileage " +str(prediction[0]) +". Go for a ride nearby."

if(output>29 and output<=46):

    pred="High performance with mileage " +str(prediction[0]) +". Go for a healthy ride"

if(output>46):

    pred="Very high performance with mileage " +str(prediction[0])+". You can plan for a Tour"

    return render_template('index.html', prediction_text='{}'.format(pred))

@app.route('/predict_api',methods=['POST'])

def predict_api():

    """

    For direct API calls through request

    """

    data = request.get_json(force=True)

    prediction = model.y_predict([np.array(list(data.values()))])

output = prediction[0]

    return jsonify(output)

if __name__ == "__main__":

    app.run(debug=True)

```

GITHUB AND DEMO LINK

Github : <https://github.com/IBM-EPBL/IBM-Project-44097-1660722012>

Demo Link :

https://drive.google.com/file/d/1DX2P0r2NB519NsGs7h42b8eKPy_jCxaT/view?usp=share_link