

MACHINE LEARNING BASED VEHICLE PERFORMANCE ANALYZER

TEAM ID:PNT2022TMID22329

**BACHELOR OF ENGINEERING
COMPUTER SCIENCE AND ENGINEERING**

**VEL TECH MULTI TECH DR.RANGARAJAN
DR.SAKUNTHALA ENGINEERING COLLEGE, AVADI**

INDUSTRY MENTOR : NIDHI

FACULTY MENTOR : SATHISH KUMAR P

Team Members:

MOHANA KRISHNAN R - 113119UG03058

NIKHIL B - 113119UG03062

SUNIL AHAMED - 113119UG03105

KANAGASABAI R - 113119UG03045

Introduction:

1.1 Project overview:

The monitoring of car performance, especially gas consumption, has so far been approached only very superficially. A typical fuel gauge, when closely monitored, shows an extremely non-linear relationship between needle movement and fuel consumption. Inaccuracies occur especially in the range of critical low fuel values of 5-10% or more. In the past, due to this limitation, some luxury cars had an audible and flashing light alarm function to indicate a low fuel condition. These systems, which add to the existing fuel level, have no more accuracy than the fuel level monitor alone.

In recent years, with the availability of computer techniques and reliable and less expensive computer equipment, a number of systems have been developed to provide some what more accurate information about vehicle performance.

1.2 PURPOSE :

The solution mainly aim on to predict the miles per gallon value based on the given input values that effect the performance of the vehicle

2.LITERATURE SURVEY

2.1 Existing problems :

predicting the performance level of cars is an important and interesting problem. The main goal of the current study is to predict the performance of the car to improve certain behavior of the vehicle. This can significantly help to improve the systems fuel consumption and increase the efficiency. The performance analysis of the car based on the engine type, no of engine cylinders, fuel type and horsepower etc. These are the factors on which the health of the car can be predicted. It is an on-going process of obtaining, researching, analyzing and recording the health based on the above three factors. The performance objectives like mileage, dependability, flexibility and cost can be grouped together to play a vital role in prediction engine and engine management system.

This approach is the very important step towards understanding the vehicles performance

2.2 References :

Singh D, Singh M., "Internet of Vehicles for Smart and Safe Driving", International Conference on Connected Vehicles and Expo (ICCVE), Shenzhen, 19 - 23 Oct., 2015. (This paper has discussed about smart transportation services in cloud (Cloud-STS) for safety and convenience. STS provide driver centric board services in the cloud networks. STS composed of Vehicle to WiFi networks (V to Wi-Fi), Vehicle to Cloud Network (V to CN), Vehicle to Vehicle (V to V), and Cloud Network to service provider (CN to SP). The idea is to utilize the (Wi-Fi enabled) Smart Highways and 3D camera enabled dash board navigation device to enhance accident prevention / monitoring and control.)

Zhang, Y., Lin, W., and Chin, Y., "Data - Driven Driving Skill Characterization: Algorithm Comparison and Decision Fusion," SAE Technical Paper 2009 -01 -1286, 2009, <https://doi.org/10.4271/2009-01-1286>. Azevedo, C. L Cardoso. (By adapting vehicle control systems to the skill level of the driver, the overall vehicle active safety provided to the driver can be further enhanced for the existing active vehicle controls, such as ABS, Traction Control, Vehicle Stability Enhancement Systems. As a follow-up to the feasibility study in, this paper provides some recent results on data-driven driving skill characterization. In particular, the paper presents an enhancement of discriminant features, the comparison of three different learning algorithms for recognizer design, and the performance enhancement with decision fusion. The paper concludes with the discussions of the experimental results and some of the future work.

J. E. Meseguer, C. T. Calafate, J. C. Cano and P. Manzoni, "Driving Styles: A smartphone application to assess driver behaviour," 2013 IEEE Symposium on Computers and Communications (ISCC), Split, 2013, pp. 000535 - 000540. doi:10.1109/ISCC.2013.6755001. (The Driving Styles architecture integrates both data mining techniques and neural networks to generate a classification of driving styles by analyzing the driver behavior along each route. In particular, based on parameters such as speed, acceleration, and revolutions per minute of the engine (rpm), we have implemented a neural network-based algorithm that is able to characterize the type of road on which the vehicle is moving, as well as the degree of aggressiveness of each driver. The final goal is to assist drivers at correcting the bad habits in their driving behaviour, while offering helpful tips to improve fuel economy. In this work we take advantage of two key-points: the evolution of mobile terminals and the availability of a standard interface to access car data).

Kenneth L. Clarkson. 1985. Algorithms for Closest - Point Problems (Computational Geometry). Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171. (This dissertation reports a variety of new algorithms for solving closest point problems. The input to these algorithms is a set or sets of points in d-dimensional space, with an associated $L(p)$ metric. The problems considered are: (1) The all nearest neighbours' problem. For point set A, find the nearest neighbours in A of each point in A. (2) The nearest foreign neighbour problem. For point sets A and B, find the closest point in B to each point in A. (3) The geometric minimum spanning tree problem).

Goszczynska H., Kowalczyk L., Kuraszkiewicz B. (2014) Correlation Matrices as a tool to Analyze the Variability of EEG Maps. In: Piętko E., Kawa J., Wieclawek W. (eds) Information Techn

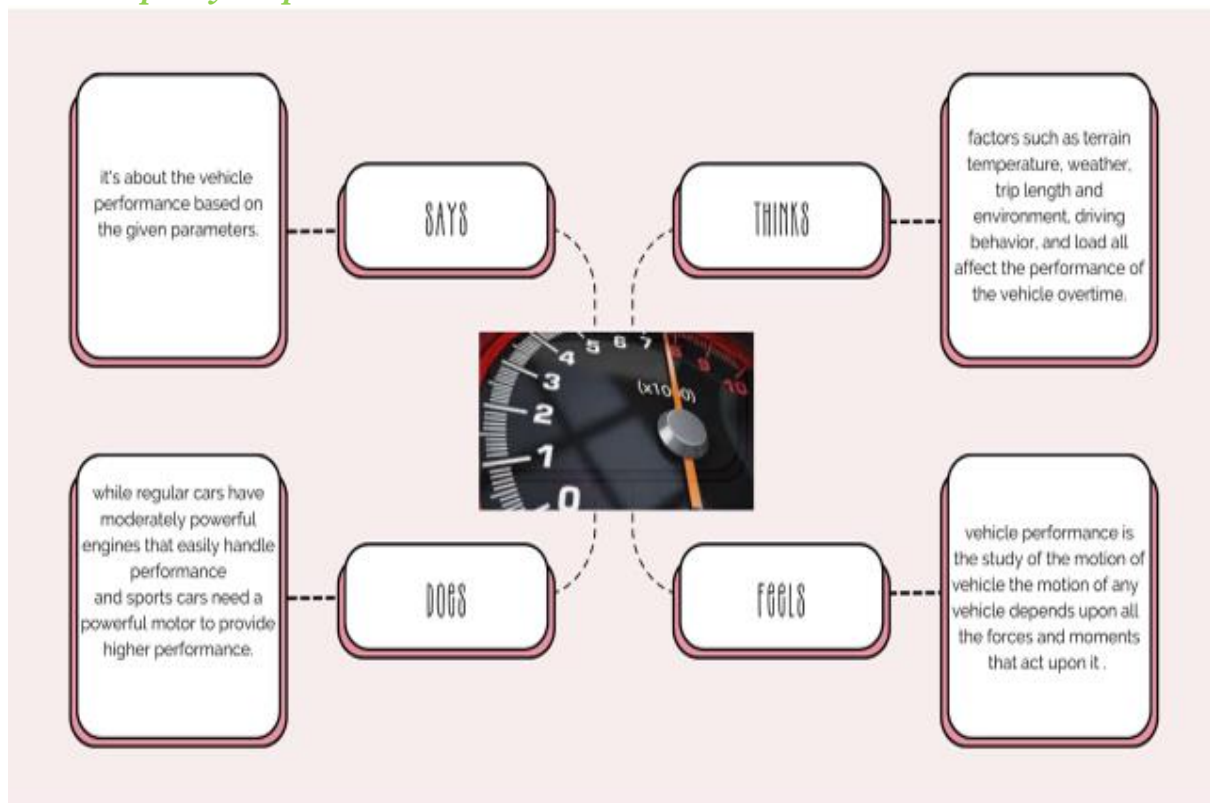
ologies in Biomedicine, Volume 4. Advances in Intelligent Systems and Computing, vol 284. Springer. (The aim of this paper is to present the selected examples of possible applications of image of correlation coefficients matrix of EEG map series in the analysis of variation of the topography of the isopotential areas in EEG maps, and thus in the assessment of stationarity, spatio-temporal variability and trends of changes of bioelectric activity of the brain. The image of correlation coefficients matrix shows similarity of all pairs of maps in a series. The choice of segmentation threshold of characteristic areas in images of the correlation coefficients matrix of EEG map series corresponding to the sequence similarity relationships in a series of maps was based on the results of research conducted on test series).

2.3 Problem statement definition:

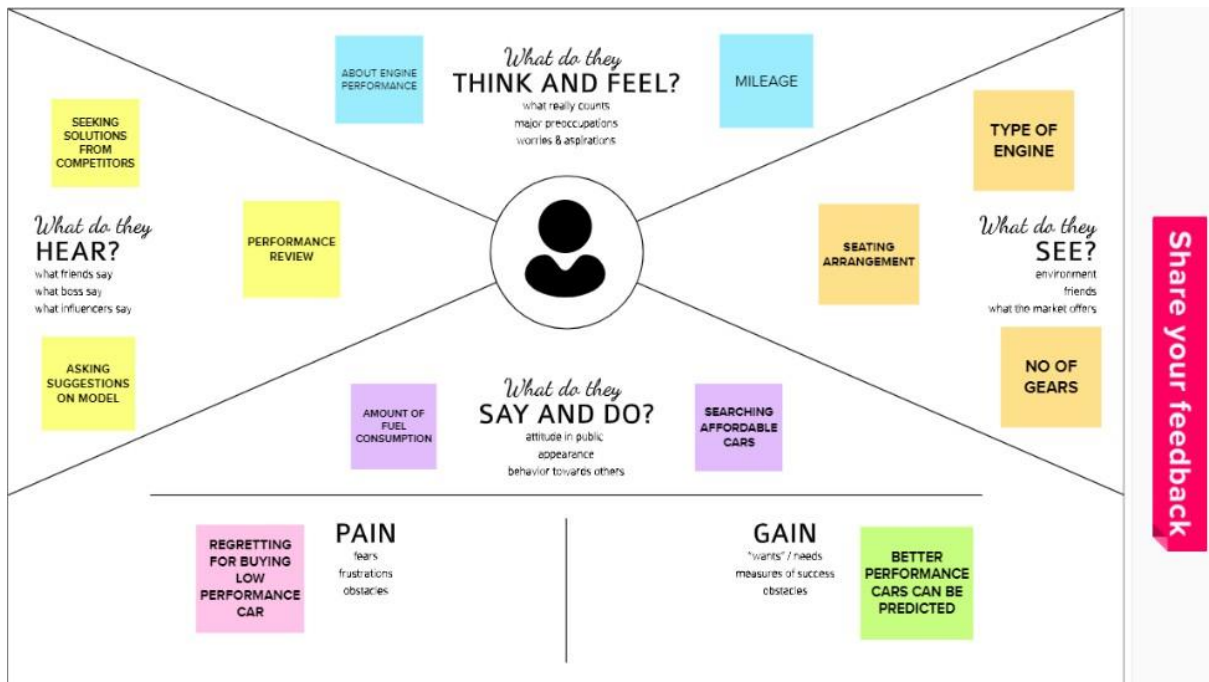
It is an important to analyse the factors using number of well-known approaches of machine learning algorithms like linear regression, decision tree and random forest to improve the vehicle performance efficiency. The range, durability and longevity of automotive traction batteries are 'hot topics' in automotive engineering. And here we consider a performance in mileage. To solve this problem, we will develop the models, using the different algorithms and neural networks. We will then see which algorithm predicts car performance (Mileage) with higher accuracy.

3. Ideation and proposed solution:

3.1 Empathy map canvas



3.2 IDEATION AND BRAIN STORMING :



3.3 PROPOSED SOLUTION

Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement	The user is an Automobile manufacturer which is looking for new technologies to analyse the performance of its vehicles.
2.	Idea / Solution description	The variables that are measured across different parts of the car and its performance can be used to forecast the condition of the vehicle. This continual process of collecting, studying, analysing, and documenting vehicle's performance can be used to gain competitive advantage.
3.	Novelty / Uniqueness	Giving the public and the manufacturer the feature to analyse their vehicle's performance.
4.	Social Impact / Customer Satisfaction	The product improves vehicles performance, increases mileage, efficiency, lifetime, etc. These properties will have a positive response.
5.	Business Model	Selling the product as an application and/or selling through collaboration with vehicle manufacturers. Product can also be sold to Racing companies for them to dominate in their field
6.	Scalability of the Solution	The project will be scalable when the parts used to measure data in vehicles is feasible and the ML model is fast in processing data.

3.4 PROPOSED SOLUTION I

1. Customer Segment Automobile Manufacturers	6. Customer Limitations Change in existing manufacturing automobile model manufacturing process. Formation of new domain for the programming, installation & maintenance of the performance analyser	5. Available Solutions Investing in some other technology - But Vehicle Performance Analyser can give their customers the satisfaction of understanding their vehicle, improving its performance, etc.
2. Problems / Pains Lack of new technology to survive in the new era of smart automobiles. Inability to assure durability and maintenance of automobiles.	9. Problem Root / Cause Increase in fuel prices --> Better mileage --> Better maintenance --> Additional support for unskilled labourers and guidance for general public.	7. Behaviour Manufacture new parts with embedded technology. Hire and form teams to program, deploy & maintain the application. Take feedbacks and improve the model performance in new models.
3. Triggers to Act Problem Solving		
Machine Learning Based Vehicle Performance Analyser		
Seeing rival companies coming up with new technology.		
Demand for smart technologies in vehicles.		
4. Emotions Imperiled --> Confident		

4. REQUIREMENT ANALYSIS :

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	<ul style="list-style-type: none">• The system doesn't require any prior technical knowledge from the user, thus even a novice user can access it.• The user interface would prioritize recognition over recall.• User friendly• Pay attention to internal sources of control• It wouldn't take long for the content to load and show (30 seconds).• The fields in the site would be self-explaining
NFR-2	Security	<ul style="list-style-type: none">• Only the authenticated user will be able to use the site's services.• The database should be backed up every hour.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Details	No of cylinders, Displacement, Horsepower, Weight, Model year, and Origin
FR-4	User Requirements	<ul style="list-style-type: none">• Upload all essential details to the website's appropriate.• The system would extract all essential data based on the uploads.• Based on the information that was scraped, a list of every potential potential results will be delivered.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

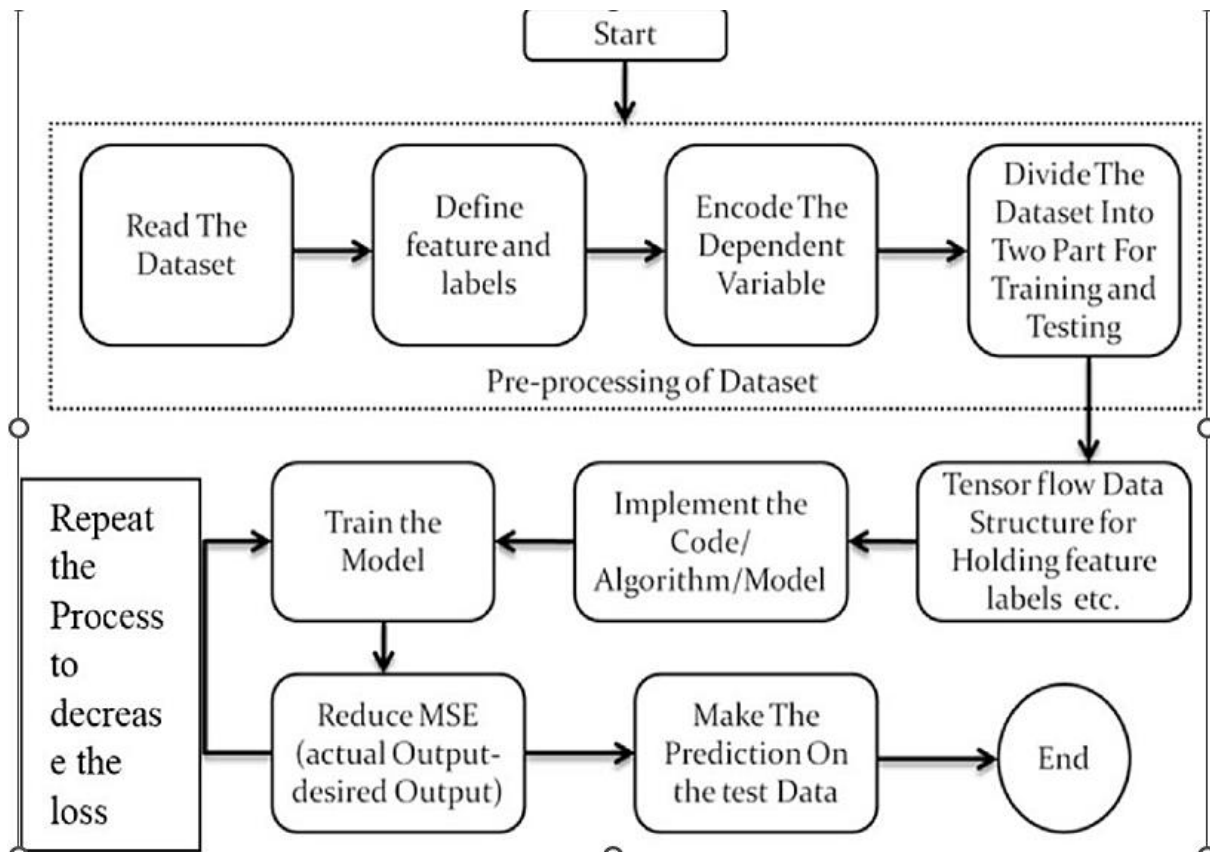
NFR-3	Reliability	<ul style="list-style-type: none">• Due to the value of data and the potential harm that inaccurate or incomplete data could do, the system will always strive for optimum reliability.• The system will be operational every day of the week, 24 hours a day.
-------	-------------	---

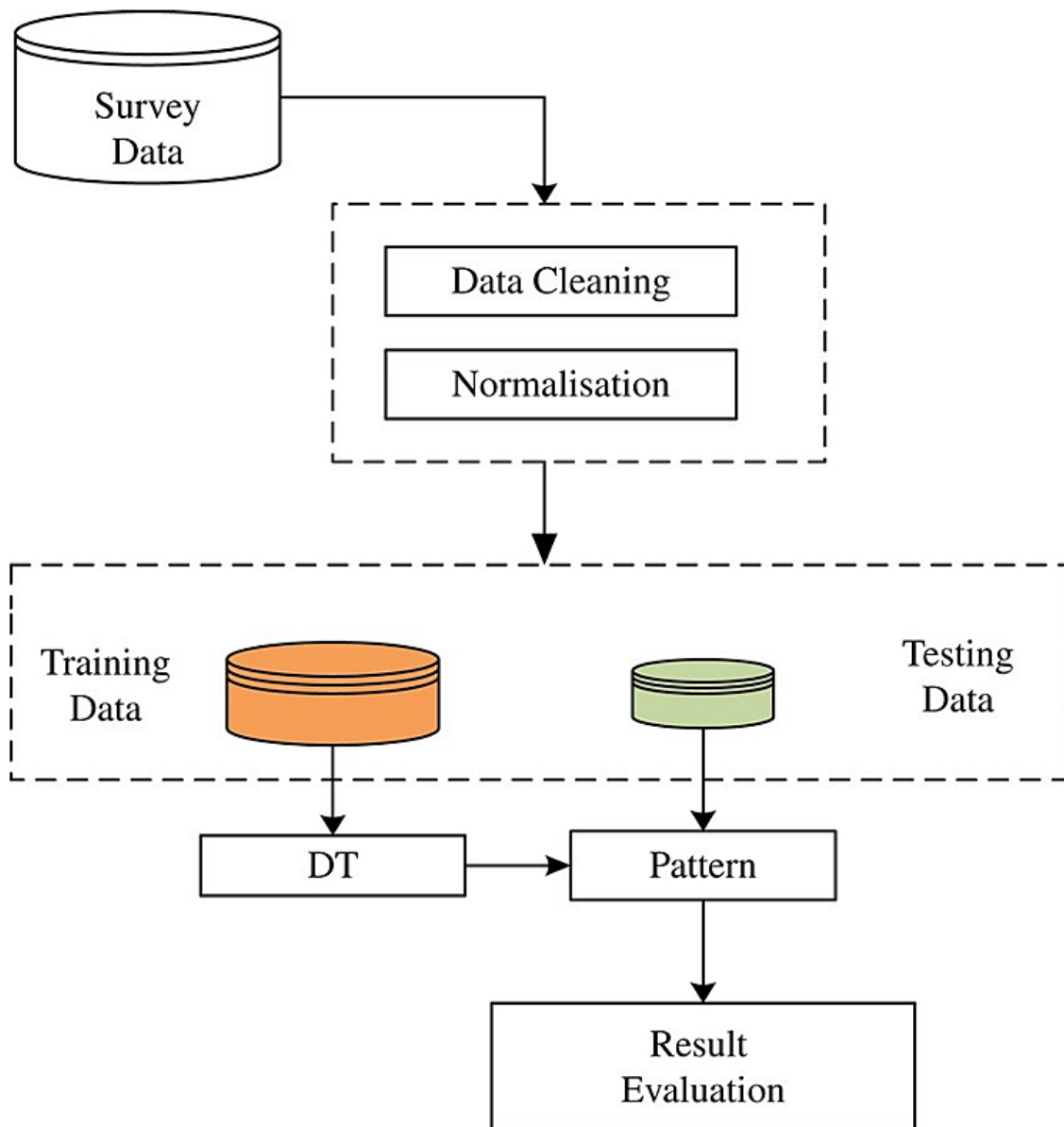
NFR-4	Performance	<ul style="list-style-type: none"> • The website can efficiently handle traffic by responding to requests right away. • A 64-kbps modem connection would take no longer than 30 seconds to see this webpage (quantitatively, the mean time)
NFR-5	Availability	<ul style="list-style-type: none"> • Low data redundancy • reduced error risk, quick and effective
NFR-6	Scalability	<ul style="list-style-type: none"> • A significant number of users must be able to access the system simultaneously because an academic portal is essential to the courses that use it. • The system will likely be most stressed during the admissions season. • Therefore, it must be able to handle several users at once.

5. **PROJECT DIAGRAM :**

5.1 **DATA FLOW DIAGRAM:**

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored





5.2 SOLUTION AND TECHNICAL ARCHITECTURE :

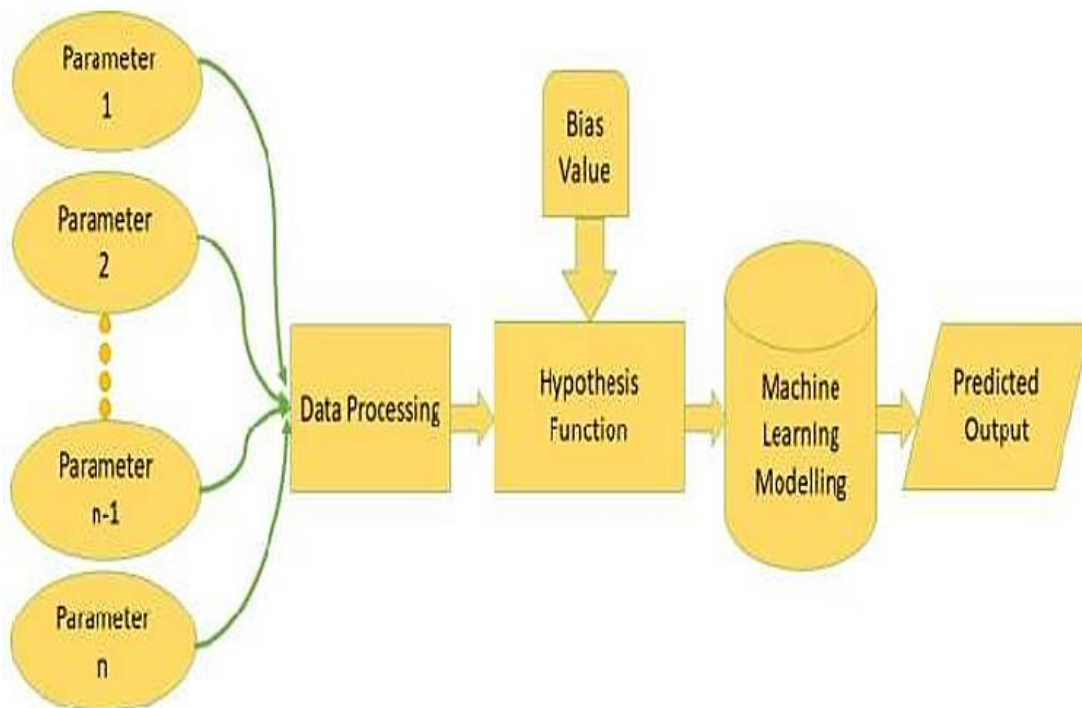
Solution:

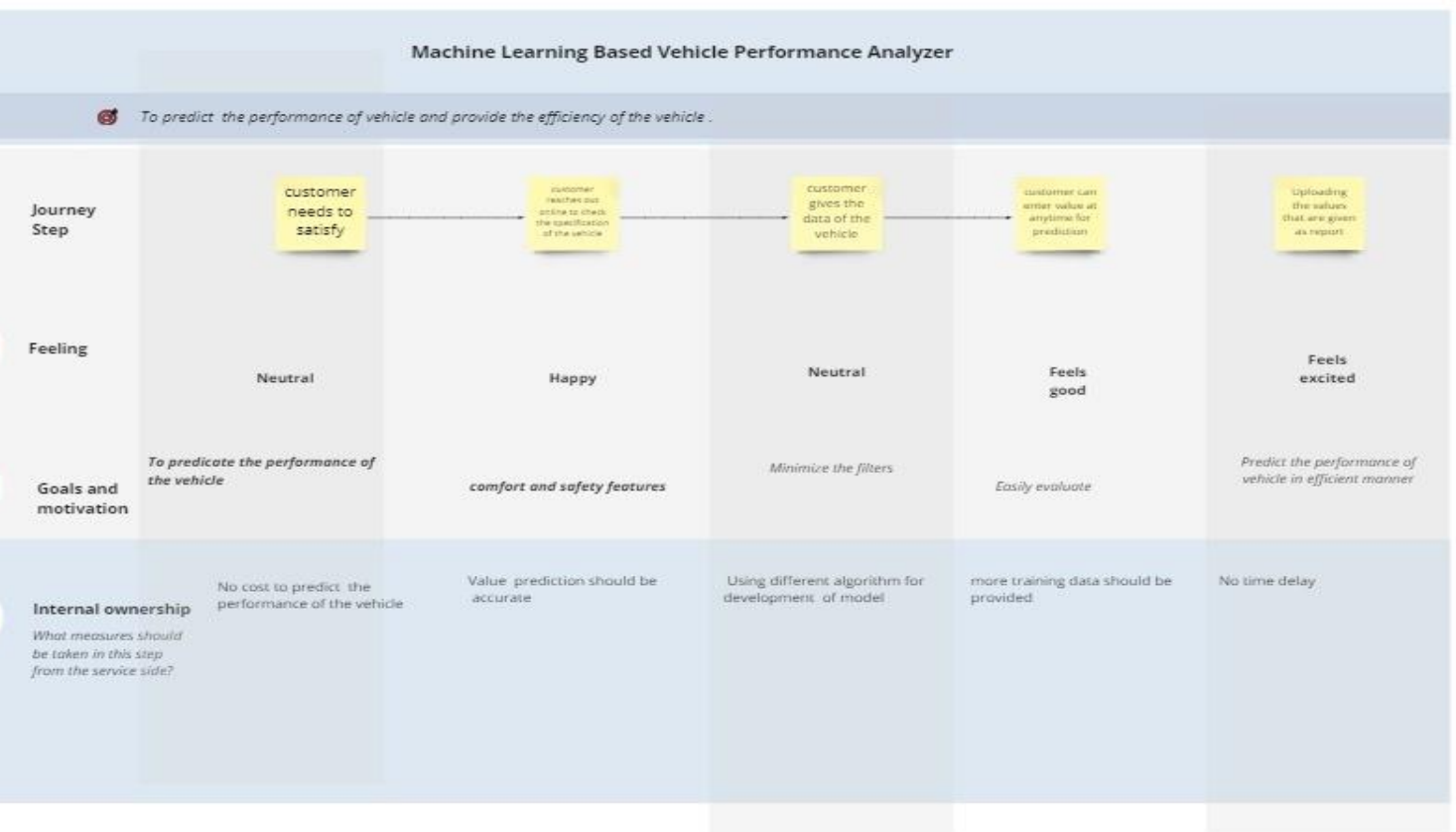
predicting the performance level of cars is an important and interesting problem. The main goal of the current study is to predict the performance of the car to improve certain behavior of the vehicle. This can significantly help to improve the systems fuel consumption and increase the efficiency. The performance analysis of the car based on the engine type, no of engine cylinders, fuel type and horsepower etc. These are the factors on which the health of the car can be predicted. It is an on-going process of obtaining, researching, analyzing

and recording the health based on the above three factors. The performance objectives like mileage, dependability, flexibility and cost can be grouped together to play a vital role in prediction engine and engine management system.

This approach is the very important step towards understanding the vehicle's performance

It is important to analyse the factors using number of well-known approaches of machine learning algorithms like linear regression, decision tree and random forest to improve the vehicle performance efficiency. The range, durability and longevity of automotive traction batteries are 'hot topics' in automotive engineering. And here we consider a performance in mileage. To solve this problem, we will develop the models, using the different algorithms and neural networks. We will then see which algorithm predicts car performance (Mileage) with higher accuracy.





5.3 USER STORIES :

6. PROJECT PLANNING AND SCHEDULING :

6.1 SPRINT PLANNING

Steps To Perform Predictive Analysis:

Some basic steps should be performed in order to perform predictive analysis.

Define Problem Statement:

Define the project outcomes, the scope of the effort, objectives, identify the datasets that are going to be used.

Data Collection:

Data collection involves gathering the necessary details required for the analysis. It involves the historical or past data from an authorized source over which predictive analysis is to be performed.

Data Cleaning:

Data Cleaning is the process in which we refine our data sets. In the process of data cleaning, we remove unnecessary and erroneous data. It involves removing the redundant data and duplicated data from our data sets.

Data Analysis:

It involves the exploration of data. We explore the data and analyze it thoroughly in order to identify some patterns or new outcomes from the data set. In this stage, we discover useful information and conclude by identifying some patterns or trends.

Build Predictive Model:

In this stage of predictive analysis, we use various algorithms to build predictive models based on the patterns observed. It requires knowledge of python, R, Statistics and MATLAB and so on. We also test our hypothesis using standard statistical models.

Validation: It is a very important step in predictive analysis. In this step, we check the efficiency of our model by performing various tests. Here we provide sample input sets to check the validity of our model. The model needs to be evaluated for its accuracy in this stage.

Deployment:

In deployment we make our model work in a real environment and it helps in everyday decision making and make it available to use.

Model Monitoring:

Regularly monitor your models to check performance and ensure that we have proper results. It is seeing how model predictions are

6.2 SPRINT DELIVERY SCHEDULE :

Project Tracker, Velocity & Burn down Chart: (4 Marks)

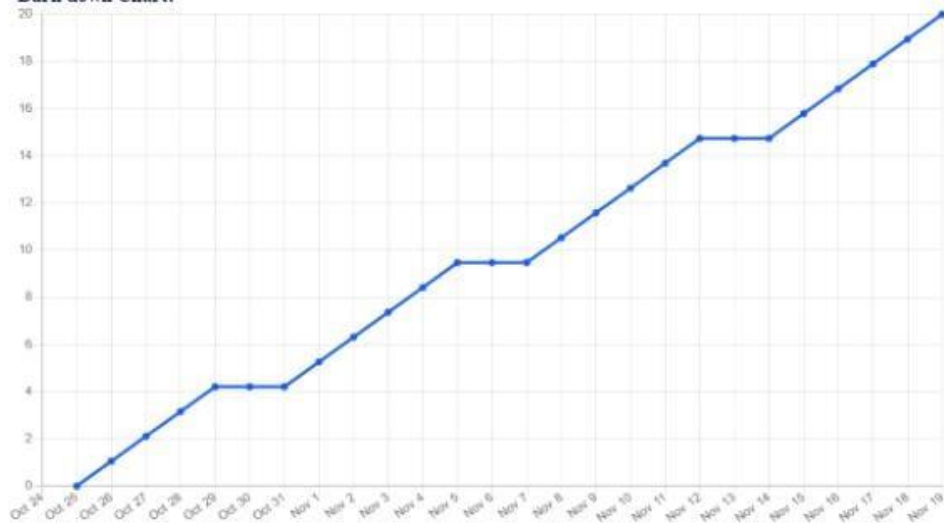
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	20 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{6} = 3.33$$

Burn down Chart:



6.3 Reports from JIRA

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected R
TC_OO1	Functional	Home Page	Verify user is able to see the homepage		1.Enter URL and click go	Homepage Url	Login/Signup popup sh
TC_OO2	UI	Home Page	Verify the UI output in submit page		1.Enter URL and click go 2.Click on submit button	Homepage Url	Application should sho elements before click button : No.of Cylinde Displacement Horsepower Weight Modal Year Origin
TC_OO3	Functional	Home page	Verify user is able to get output		1.Enter URL and click go 2.Enter the vehicle parimeters for the prediction	No.of Cylinders : 8 Displacement : 110 Horsepower : 210 Weight : 3246 Modal Year : 70 Origin : 3	User should navigate t
TC_OO4	Functional	Submit page	Verify user is able to get output with InValid credentials		1.Enter URL and click go 2.Enter the vehicle parimeters for the prediction 3.Then click on the submit button	No.of Cylinders : 8 Displacement : 110 Modal Year : 70 Origin : 3	Application should sho and 'horsepower' valid
TC_OO4	Functional	Submit page	Verify user is able to get output with InValid credentials		1.Enter URL and click go 2.Enter the vehicle parimeters for the prediction 3.Then click on the submit button 4. Output page is visible with predicted value of the performance	No.of Cylinders : 8 Displacement : 110 Horsepower : 210 Weight : 3246 Modal Year : 70	Application should sho validation message.
					1.Enter URL and click go	Displacement : 110	Application should sho

7.CODING & SOLUTIONING :

Feature 1 :

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

Importing Dataset

```
dataset=pd.read_csv('car performance.csv')
dataset
```

	mpg	cylinders	displacement	horsepower	weight	
acceleration \						
0	18.0	8	307.0	130.0	3504	12.0
1	15.0	8	350.0	165.0	3693	11.5
2	18.0	8	318.0	150.0	3436	11.0
3	16.0	8	304.0	150.0	3433	12.0
4	17.0	8	302.0	140.0	3449	10.5
..
393	27.0	4	140.0	86.0	2790	15.6
394	44.0	4	97.0	52.0	2130	24.6
395	32.0	4	135.0	84.0	2295	11.6
396	28.0	4	120.0	79.0	2625	18.6
397	31.0	4	119.0	82.0	2720	19.4

	model	year	origin	car name
0		70	1	chevrolet chevelle malibu
1		70	1	buick skylark 320
2		70	1	plymouth satellite
3		70	1	amc rebel sst
4		70	1	ford torino
..
393		82	1	ford mustang gl
394		82	2	vw pickup

```

395      82      1      dodge rampage
396      82      1      ford ranger
397      82      1      chevy s-10

```

```
[398 rows x 9 columns]
```

Finding missing data

```
dataset.isnull().any()
```

```

mpg      False
cylinders False
displacement False
horsepower True
weight   False
acceleration False
model year False
origin   False
car name  False
dtype: bool

```

There are no null characters in the columns but there is a special character '?' in the 'horsepower' column. So we replaced '?' with nan and replaced nan values with mean of the column.

```
dataset['horsepower']=dataset['horsepower'].replace('?',np.nan)
```

```
dataset['horsepower'].isnull().sum()
```

```
6
```

```
dataset['horsepower']=dataset['horsepower'].astype('float64')
```

```
dataset['horsepower'].fillna((dataset['horsepower'].mean()),inplace=True)
```

```
dataset.isnull().any()
```

```

mpg      False
cylinders False
displacement False
horsepower False
weight   False
acceleration False
model year False
origin   False
car name  False
dtype: bool

```

`dataset.info()` *#Pandas dataframe.info() function is used to get a quick overview of the dataset.*

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   mpg              398 non-null    float64
1   cylinders        398 non-null    int64
2   displacement     398 non-null    float64
3   horsepower       398 non-null    float64
4   weight           398 non-null    int64
5   acceleration     398 non-null    float64
6   model year      398 non-null    int64
7   origin           398 non-null    int64
8   car name        398 non-null    object
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB

```

`dataset.describe()` *#Pandas describe() is used to view some basic statistical details of a data frame or a series of numeric values.*

	mpg	cylinders	displacement	horsepower	
weight \ count	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623
std	7.815984	1.701004	104.269838	38.199187	846.841774
min	9.000000	3.000000	68.000000	46.000000	1613.000000
25%	17.500000	4.000000	104.250000	76.000000	2223.750000
50%	23.000000	4.000000	148.500000	95.000000	2803.500000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000

	acceleration	model year	origin
count	398.000000	398.000000	398.000000
mean	15.568090	76.010050	1.572864
std	2.757689	3.697627	0.802055
min	8.000000	70.000000	1.000000
25%	13.825000	73.000000	1.000000
50%	15.500000	76.000000	1.000000
75%	17.175000	79.000000	2.000000
max	24.800000	82.000000	3.000000

There is no use with car name attribute so drop it

```
dataset=dataset.drop('car name',axis=1) #dropping the unwanted column.
corr_table=dataset.corr()#Pandas dataframe.corr() is used to find the
pairwise correlation of all columns in the dataframe.
corr_table
```

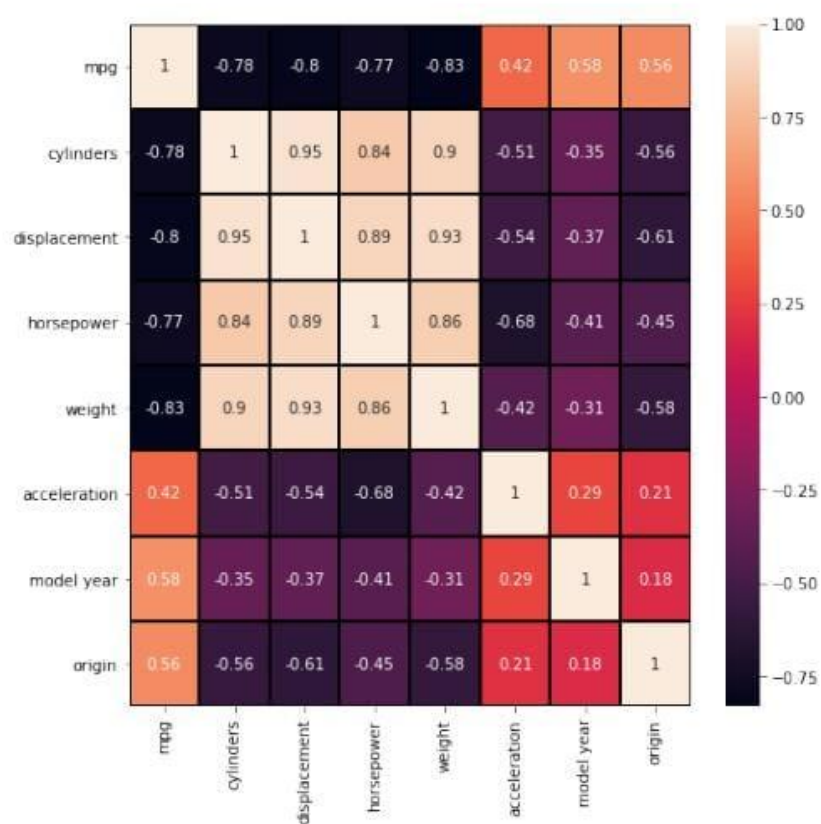
\	mpg	cylinders	displacement	horsepower	weight
mpg	1.000000	-0.775396	-0.804203	-0.771437	-0.831741
cylinders	-0.775396	1.000000	0.950721	0.838939	0.896017
displacement	-0.804203	0.950721	1.000000	0.893646	0.932824
horsepower	-0.771437	0.838939	0.893646	1.000000	0.860574
weight	-0.831741	0.896017	0.932824	0.860574	1.000000
acceleration	0.420289	-0.505419	-0.543684	-0.684259	-0.417457
model year	0.579267	-0.348746	-0.370164	-0.411651	-0.306564
origin	0.563450	-0.562543	-0.609409	-0.453669	-0.581024

	acceleration	model year	origin
mpg	0.420289	0.579267	0.563450
cylinders	-0.505419	-0.348746	-0.562543
displacement	-0.543684	-0.370164	-0.609409
horsepower	-0.684259	-0.411651	-0.453669
weight	-0.417457	-0.306564	-0.581024
acceleration	1.000000	0.288137	0.205873
model year	0.288137	1.000000	0.180662
origin	0.205873	0.180662	1.000000

Data Visualizations

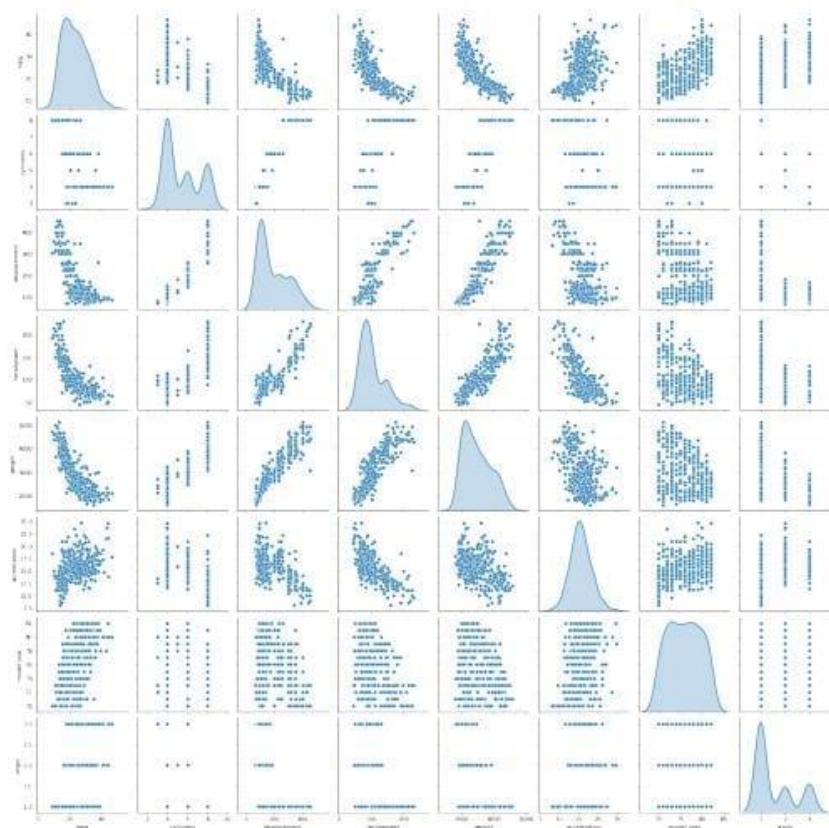
Heatmap : which represents correlation between attributes

```
sns.heatmap(dataset.corr(),annot=True,linewidth='black', linewidths =
1)#Heatmap is a way to show some sort of matrix plot,annot is used for
correlation.
fig=plt.gcf()
fig.set_size_inches(8,8)
```



Visualizations of each attributes w.r.t rest of all attributes

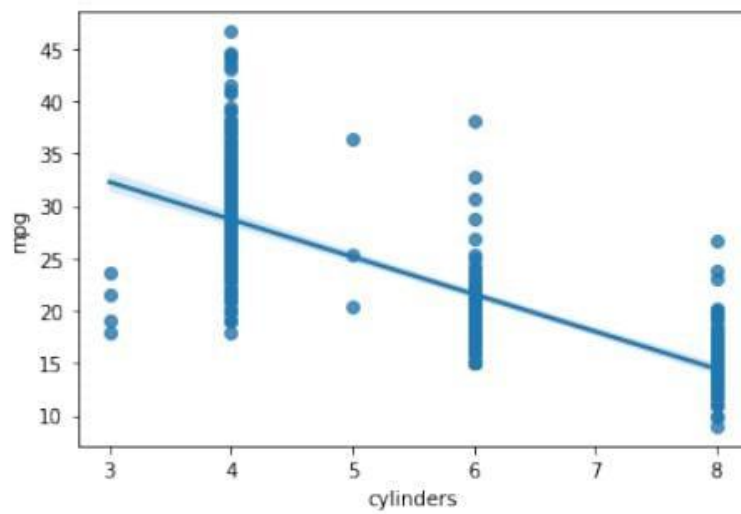
```
sns.pairplot(dataset,diag_kind='kde') #pairplot represents pairwise
relation across the entire dataframe.
plt.show()
```



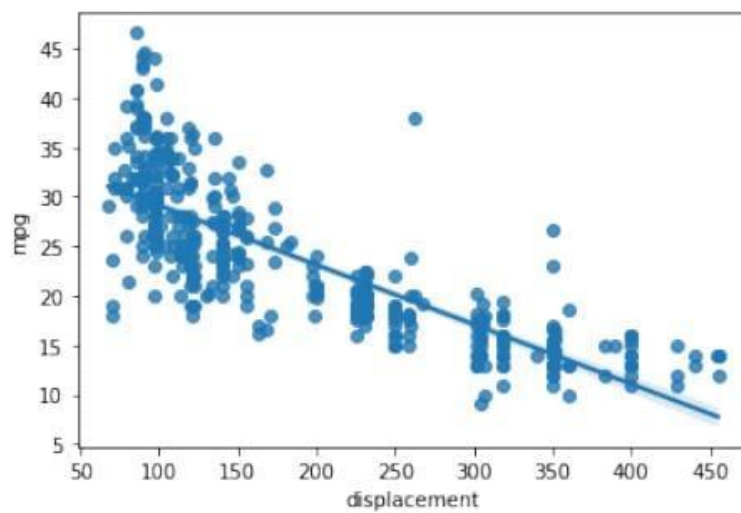
Regression plots(`regplot()`) creates a regression line between 2 parameters and helps to visualize their linear relationships.

```
sns.regplot(x="cylinders", y="mpg", data=dataset)
```

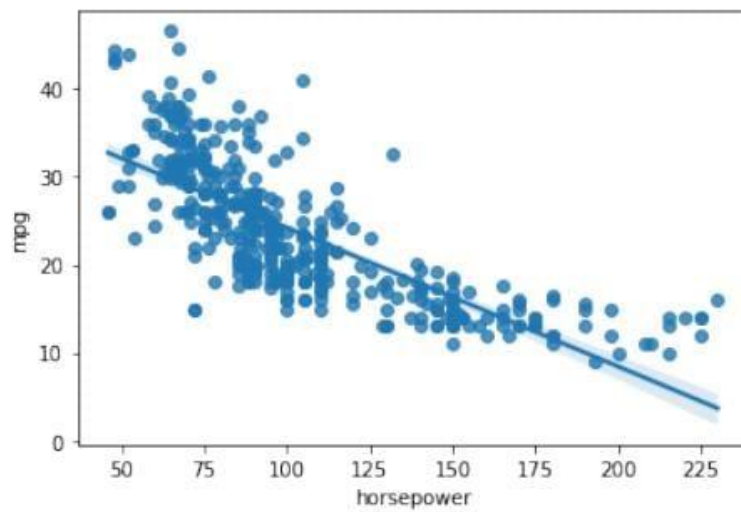
```
<matplotlib.axes._subplots.AxesSubplot at 0x279da5b95c8>
```

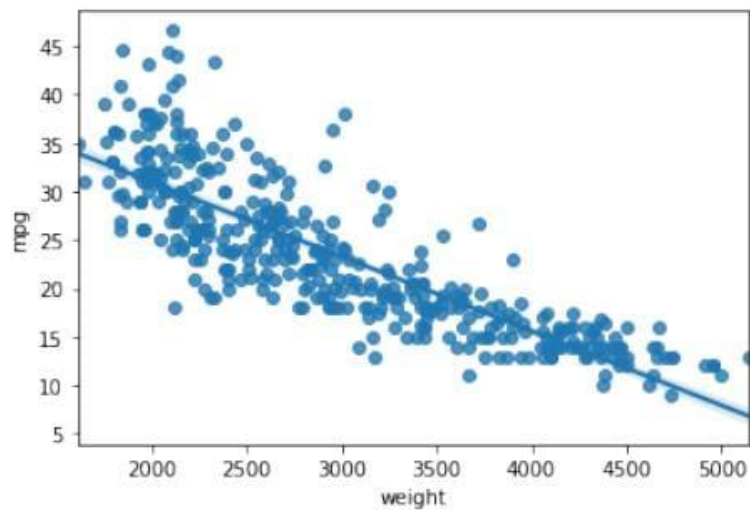
```
sns.regplot(x="displacement", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279dbd8ed08>
```



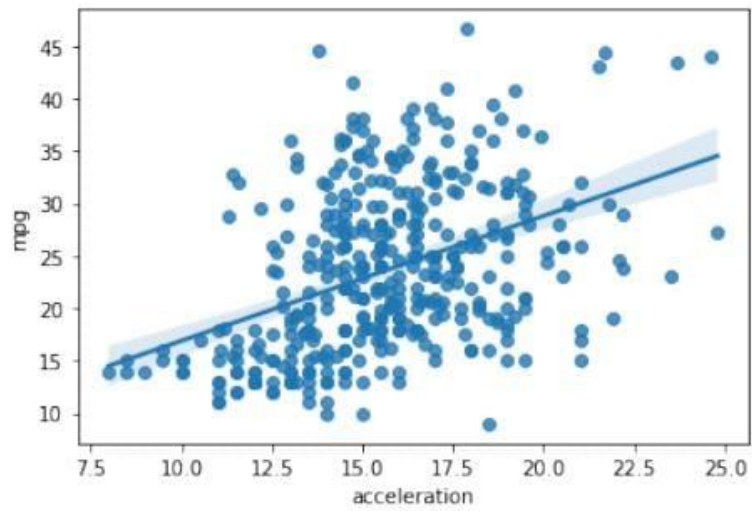
```
sns.regplot(x="horsepower", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279daab38c8>
```



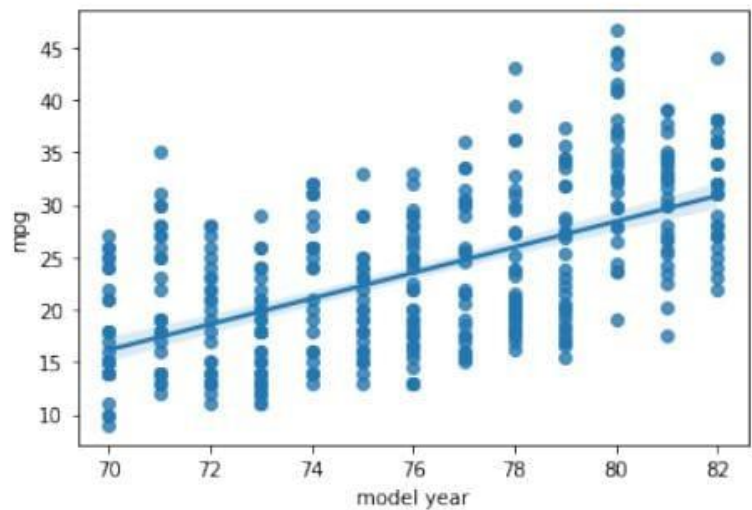
```
sns.regplot(x="weight", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279dc683588>
```



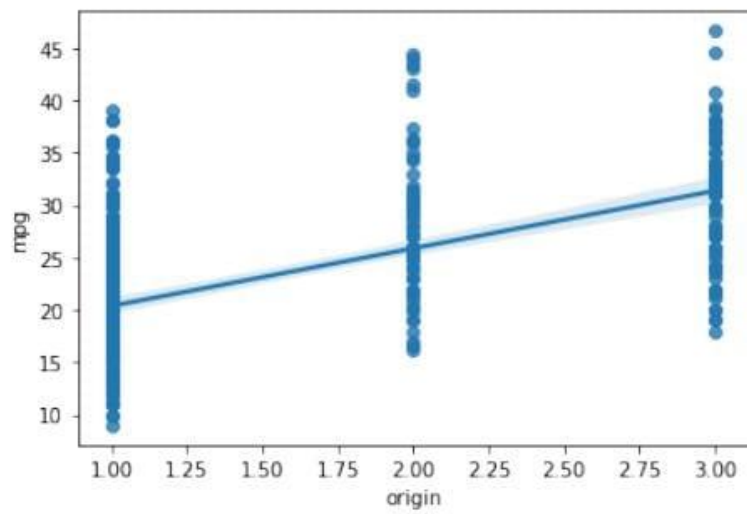
```
sns.regplot(x="acceleration", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279dc6f7c88>
```



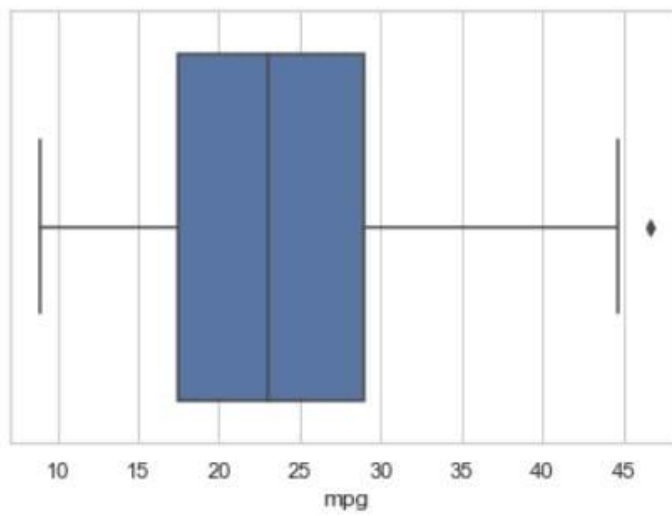
```
sns.regplot(x="model year", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279dc6e6c88>
```



```
sns.regplot(x="origin", y="mpg", data=dataset)  
<matplotlib.axes._subplots.AxesSubplot at 0x279dc7eabc8>
```



```
sns.set(style="whitegrid")  
sns.boxplot(x=dataset["mpg"])  
<matplotlib.axes._subplots.AxesSubplot at 0x279dc85a588>
```



Finding quartiles for mpg

The P-value is the probability value that the correlation between these two variables is statistically significant.

Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

By convention, when the p-value is ≤ 0.001 : we say there is strong evidence that the correlation is significant. the p-value is ≤ 0.05 : there is moderate evidence that the correlation is significant. the p-value is ≤ 0.1 : there is weak evidence that the correlation is significant. the p-value is ≤ 0.1 : there is no evidence that the correlation is significant.

```
from scipy import stats
```

Let's calculate the Pearson Correlation Coefficient and P-value of 'Cylinders' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['cylinders'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is -0.7753962854205543 with a P-value of P = 4.503992246176927e-81

Since the p-value is ≤ 0.001 , the correlation between cylinders and mpg is statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.

Let's calculate the Pearson Correlation Coefficient and P-value of 'Displacement' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['displacement'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is -0.804202824805898 with a P-value of P = 1.655888910192639e-91

Let's calculate the Pearson Correlation Coefficient and P-value of 'horsepower' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['horsepower'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is -0.7714371350025524 with a P-value of P = 9.255477533169094e-80

Since the p-value is ≤ 0.001 , the correlation between horsepower and mpg is statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative and moderately strong.

Let's calculate the Pearson Correlation Coefficient and P-value of 'weight' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['weight'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is -0.831740933244335 with a P-value of P = 2.9727995640500577e-103

Let's calculate the Pearson Correlation Coefficient and P-value of 'Acceleration' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['acceleration'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is 0.42028891210165065 with a P-value of P = 1.823091535078553e-18

Let's calculate the Pearson Correlation Coefficient and P-value of 'Model year' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['model year'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is 0.5792671330833096 with a P-value of P = 4.844935813365483e-37

Let's calculate the Pearson Correlation Coefficient and P-value of 'Origin' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['origin'],
dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

The Pearson Correlation Coefficient is 0.5634503597738432 with a P-value of P = 1.0114822102335907e-34

Ordinary Least Squares Statistics

```
test=smf.ols('mpg~cylinders+displacement+horsepower+weight+acceleration+origin',dataset).fit()
test.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

OLS Regression Results

```
=====
Dep. Variable:          mpg    R-squared:
```

```

0.717
Model: OLS Adj. R-squared:
0.713
Method: Least Squares F-statistic:
165.5
Date: Mon, 27 Apr 2020 Prob (F-statistic):
4.84e-104
Time: 10:47:17 Log-Likelihood:
-1131.1
No. Observations: 398 AIC:
2276.
Df Residuals: 391 BIC:
2304.
Df Model: 6

```

Covariance Type: nonrobust

```

=====
=====
coef      std err      t      P>|t|      [0.025
0.975]
-----
-----
Intercept  42.7111    2.693    15.861    0.000    37.417
48.005
cylinders  -0.5256    0.404    -1.302    0.194    -1.320
0.268
displacement  0.0106    0.009     1.133    0.258    -0.008
0.029
horsepower  -0.0529    0.016    -3.277    0.001    -0.085
-0.021
weight      -0.0051    0.001    -6.441    0.000    -0.007
-0.004
acceleration  0.0043    0.120     0.036    0.972    -0.232
0.241
origin      1.4269    0.345     4.136    0.000     0.749
2.105
=====
=====
Omnibus:      32.659  Durbin-Watson:
0.886
Prob(Omnibus): 0.000  Jarque-Bera (JB):
43.338
Skew:         0.624  Prob(JB):
3.88e-10
Kurtosis:     4.028  Cond. No.
3.99e+04
=====
=====

```

```

0.717
Model: OLS Adj. R-squared:
0.713
Method: Least Squares F-statistic:
165.5
Date: Mon, 27 Apr 2020 Prob (F-statistic):
4.84e-104
Time: 10:47:17 Log-Likelihood:
-1131.1
No. Observations: 398 AIC:
2276.
Df Residuals: 391 BIC:
2304.
Df Model: 6

```

Covariance Type: nonrobust

```

=====
=====
coef      std err      t      P>|t|      [0.025
0.975]
-----
-----
Intercept  42.7111    2.693    15.861    0.000    37.417
48.005
cylinders  -0.5256     0.404    -1.302    0.194    -1.320
0.268
displacement  0.0106     0.009     1.133    0.258    -0.008
0.029
horsepower  -0.0529     0.016    -3.277    0.001    -0.085
-0.021
weight      -0.0051     0.001    -6.441    0.000    -0.007
-0.004
acceleration  0.0043     0.120     0.036    0.972    -0.232
0.241
origin      1.4269     0.345     4.136    0.000     0.749
2.105
=====
=====
Omnibus:      32.659  Durbin-Watson:
0.886
Prob(Omnibus): 0.000  Jarque-Bera (JB):
43.338
Skew:         0.624  Prob(JB):
3.88e-10
Kurtosis:     4.028  Cond. No.
3.99e+04
=====
=====

```

[15.],
[14.],
[24.],
[22.],
[18.],
[21.],
[27.],
[26.],
[25.],
[24.],
[25.],
[26.],
[21.],
[10.],
[10.],
[11.],
[9.],
[27.],
[28.],
[25.],
[25.],
[19.],
[16.],
[17.],
[19.],
[18.],
[14.],
[14.],
[14.],
[14.],
[12.],
[13.],
[13.],
[18.],
[22.],
[19.],
[18.],
[23.],
[28.],
[30.],
[30.],
[31.],
[35.],
[27.],
[26.],
[24.],
[25.],
[23.],
[20.],
[21.],

[13.],
[14.],
[15.],
[14.],
[17.],
[11.],
[13.],
[12.],
[13.],
[19.],
[15.],
[13.],
[13.],
[14.],
[18.],
[22.],
[21.],
[26.],
[22.],
[28.],
[23.],
[28.],
[27.],
[13.],
[14.],
[13.],
[14.],
[15.],
[12.],
[13.],
[13.],
[14.],
[13.],
[12.],
[13.],
[18.],
[16.],
[18.],
[18.],
[23.],
[26.],
[11.],
[12.],
[13.],
[12.],
[18.],
[20.],
[21.],
[22.],
[18.],

[19.],
[21.],
[26.],
[15.],
[16.],
[29.],
[24.],
[20.],
[19.],
[15.],
[24.],
[20.],
[11.],
[20.],
[21.],
[19.],
[15.],
[31.],
[26.],
[32.],
[25.],
[16.],
[16.],
[18.],
[16.],
[13.],
[14.],
[14.],
[14.],
[29.],
[26.],
[26.],
[31.],
[32.],
[28.],
[24.],
[26.],
[24.],
[26.],
[31.],
[19.],
[18.],
[15.],
[15.],
[16.],
[15.],
[16.],
[14.],
[17.],
[16.],

[15.],
[18.],
[21.],
[20.],
[13.],
[29.],
[23.],
[20.],
[23.],
[24.],
[25.],
[24.],
[18.],
[29.],
[19.],
[23.],
[23.],
[22.],
[25.],
[33.],
[28.],
[25.],
[25.],
[26.],
[27.],
[17.5],
[16.],
[15.5],
[14.5],
[22.],
[22.],
[24.],
[22.5],
[29.],
[24.5],
[29.],
[33.],
[20.],
[18.],
[18.5],
[17.5],
[29.5],
[32.],
[28.],
[26.5],
[20.],
[13.],
[19.],
[19.],
[16.5],

[16.5],
[13.],
[13.],
[13.],
[31.5],
[30.],
[36.],
[25.5],
[33.5],
[17.5],
[17.],
[15.5],
[15.],
[17.5],
[20.5],
[19.],
[18.5],
[16.],
[15.5],
[15.5],
[16.],
[29.],
[24.5],
[26.],
[25.5],
[30.5],
[33.5],
[30.],
[30.5],
[22.],
[21.5],
[21.5],
[43.1],
[36.1],
[32.8],
[39.4],
[36.1],
[19.9],
[19.4],
[20.2],
[19.2],
[20.5],
[20.2],
[25.1],
[20.5],
[19.4],
[20.6],
[20.8],
[18.6],
[18.1],

[19.2],
[17.7],
[18.1],
[17.5],
[30.],
[27.5],
[27.2],
[30.9],
[21.1],
[23.2],
[23.8],
[23.9],
[20.3],
[17.],
[21.6],
[16.2],
[31.5],
[29.5],
[21.5],
[19.8],
[22.3],
[20.2],
[20.6],
[17.],
[17.6],
[16.5],
[18.2],
[16.9],
[15.5],
[19.2],
[18.5],
[31.9],
[34.1],
[35.7],
[27.4],
[25.4],
[23.],
[27.2],
[23.9],
[34.2],
[34.5],
[31.8],
[37.3],
[28.4],
[28.8],
[26.8],
[33.5],
[41.5],
[38.1],
[32.1],

[37.2],
[28.],
[26.4],
[24.3],
[19.1],
[34.3],
[29.8],
[31.3],
[37.],
[32.2],
[46.6],
[27.9],
[40.8],
[44.3],
[43.4],
[36.4],
[30.],
[44.6],
[40.9],
[33.8],
[29.8],
[32.7],
[23.7],
[35.],
[23.6],
[32.4],
[27.2],
[26.6],
[25.8],
[23.5],
[30.],
[39.1],
[39.],
[35.1],
[32.3],
[37.],
[37.7],
[34.1],
[34.7],
[34.4],
[29.9],
[33.],
[34.5],
[33.7],
[32.4],
[32.9],
[31.6],
[28.1],
[30.7],
[25.4],

```
[24.2],
[22.4],
[26.6],
[20.2],
[17.6],
[28. ],
[27. ],
[34. ],
[31. ],
[29. ],
[27. ],
[24. ],
[23. ],
[36. ],
[37. ],
[31. ],
[38. ],
[36. ],
[36. ],
[36. ],
[34. ],
[38. ],
[32. ],
[38. ],
[25. ],
[38. ],
[26. ],
[22. ],
[32. ],
[36. ],
[27. ],
[27. ],
[44. ],
[32. ],
[28. ],
[31. ]])
```

Splitting into train and test data.

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
```

we are splitting as 90% train data and 10% test data

random forest regressor

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf=
RandomForestRegressor(n_estimators=10,random_state=0,criterion='mae')
rf.fit(x_train,y_train)
```

C:\Users\91938\anaconda3\lib\site-packages\ipykernel_launcher.py:2:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mae',
                       max_depth=None, max_features='auto',
max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2,
min_weight_fraction_leaf=0.0,
                       n_estimators=10, n_jobs=None, oob_score=False,
                       random_state=0, verbose=0, warm_start=False)
```

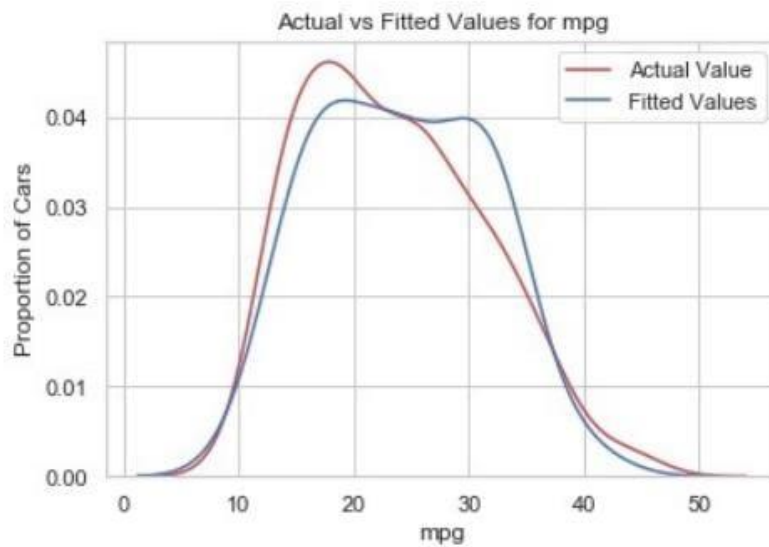
```
y_pred2=rf.predict(x_test)
y_pred2
```

```
array([14.05, 25.55, 13.7 , 21.2 , 18.5 , 30.8 , 34.31, 22.5 , 15.1 ,
       24.46, 32.01, 39.79, 17.8 , 24.85, 15.85, 31.31, 28.32, 26.93,
       16.54, 32.12, 16.05, 25.6 , 23.86, 20.56, 32.19, 24.75, 32.39,
       32.64, 31.02, 16.39, 18.35, 30.1 , 17.67, 31. , 22.91, 23.5 ,
       19.77, 16.1 , 33.65, 12.  ])
```

```
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r",
label="Actual Value")
sns.distplot(y_pred2, hist=False, color="b", label="Fitted Values" ,
ax=ax1)
```

```
plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')
```

```
plt.show()
plt.close()
```

We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
r2_score(y_test, y_pred2)
```

```
0.9013457876319049
```

```
mean_squared_error(y_test, y_pred2)
```

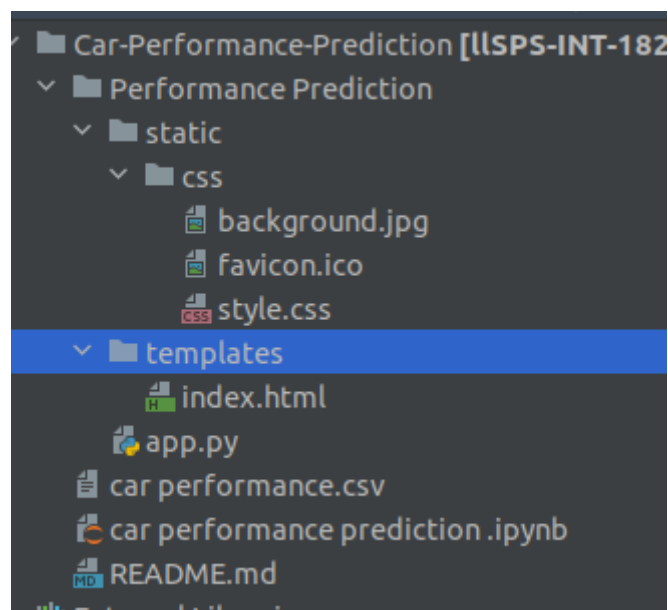
```
6.818917500000002
```

```
np.sqrt(mean_squared_error(y_test, y_pred2))
```

```
2.611305707878724
```

Feature 2 :

Application Building :



Homepage HTML code :

```
WhatsApp File Edit View Chat Call Window Help

Machine learning based vehicle performance analyzer - index.html

Project: README.md car performance prediction .ipynb index.html style.css app.py car performance (1).csv

1 <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">
2 <link href="https://fonts.googleapis.com/css2?family=Barassol&display=swap" rel="stylesheet">
3 <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
4 <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
5 <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
6 <link rel="shortcut icon" href="{{ url_for('static', filename='css/favicon.ico') }}">
7 <div class="navbar">
8   <section class="title">
9     <h1 style="text-align: center;">PREDICT YOUR CAR'S PERFORMANCE</h1>
10   </section>
11 </div>
12
13
14 <div class="wrapper fadeInDown">
15   <div id="formContent">
16     <!-- Tabs Titles -->
17     <section class="date">
18       <!-- Icon -->
19       <div class="fadeIn first">
20         <script src="https://unpkg.com/@lottiefiles/lottie-player@latest/dist/lottie-player.js"></script>
21         <lottie-player
22           src="https://assets6.lottiefiles.com/packages/lf28_TkGfat.json"
23           background="transparent"
24           speed="1"
25           loop
26           style="width: 100%; height: 100%;"
27           autoplay
28         ></lottie-player>
29       </div>
30       <div class="fadeInDown">
31         <form action="{{ url_for('y_predict') }}" method="post">
32           <input type="text" name="Cylinders" placeholder="No. of cylinders (count)" required="required" />
33           <input type="text" name="Displacement" placeholder="Displacement (in miles)" required="required" />
34         </form>
35       </div>
36     </section>
37   </div>
38 </div>

Version Control Run Python Packages TODO Python Console Problems Terminal Services
Packages installed successfully: Installed packages: 'requests' (today 8:43 pm) 14:33 CRLF UTF-8 4 spaces Python 3.10
```

```
Machine learning based vehicle performance analyzer - index.html

Project: README.md car performance prediction .ipynb index.html style.css app.py car performance (1).csv

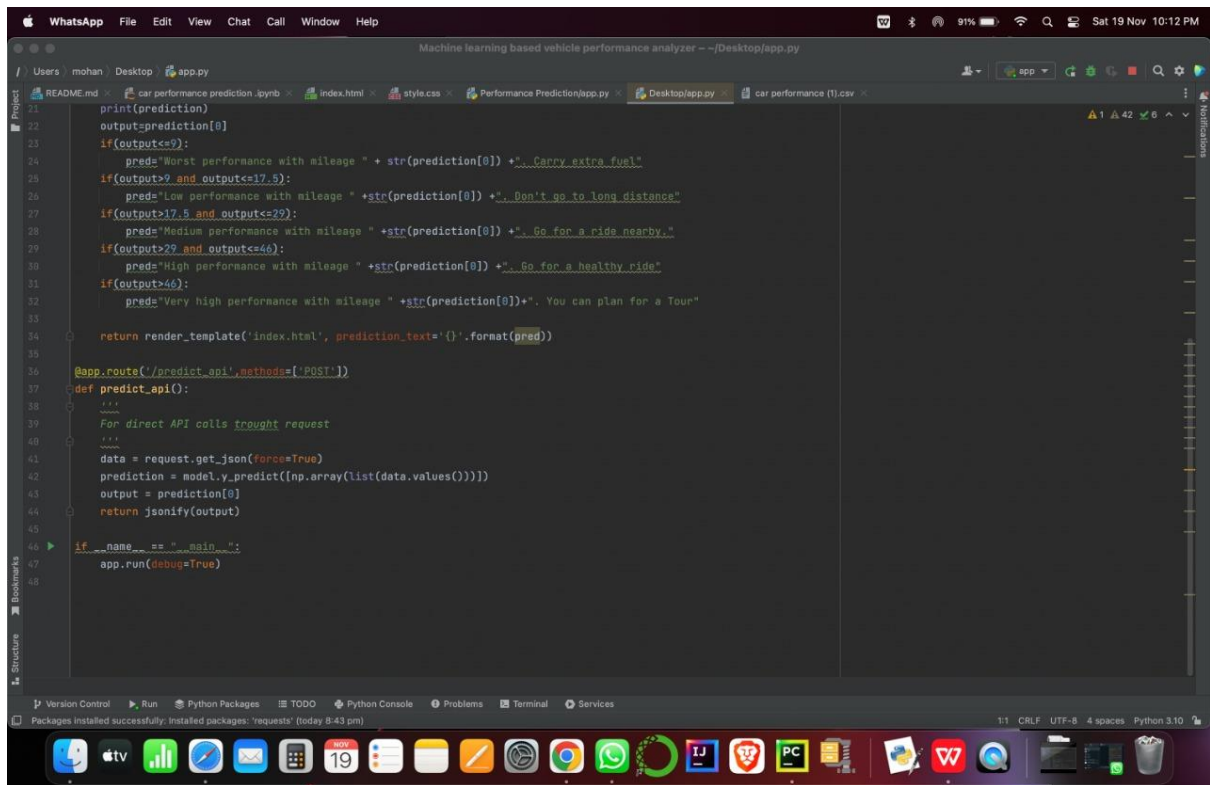
19 <div class="fadeIn first">
20   <script src="https://unpkg.com/@lottiefiles/lottie-player@latest/dist/lottie-player.js"></script>
21   <lottie-player
22     src="https://assets6.lottiefiles.com/packages/lf28_TkGfat.json"
23     background="transparent"
24     speed="1"
25     loop
26     style="width: 100%; height: 100%;"
27     autoplay
28   ></lottie-player>
29 </div>
30 <div class="fadeInDown">
31   <form action="{{ url_for('y_predict') }}" method="post">
32     <input type="text" name="Cylinders" placeholder="No. of cylinders (count)" required="required" />
33     <input type="text" name="Displacement" placeholder="Displacement (in miles)" required="required" />
34     <input type="text" name="Horsepower" placeholder="Horsepower (per sec)" required="required" />
35     <input type="text" name="Weight" placeholder="Weight (in pounds)" required="required" />
36     <input type="text" name="Model Year" placeholder="Model Year (YY)" required="required" />
37     <input type="text" name="Origin" placeholder="Origin" required="required" />
38     <br>
39     <input type="submit" class="fadeIn fourth" value="Predict">
40   </form>
41 </div>
42
43 <div id="formFooter">
44   <a class="underlineHover" href="#">
45     <strong>{{ prediction_text }}</strong></a>
46   </div>
47 </div>
48 </div>
49 </div>
50

div.wrapper.fadeInDown

Version Control Run Python Packages TODO Python Console Problems Terminal Services
Packages installed successfully: Installed packages: 'requests' (today 8:43 pm) 14:33 CRLF UTF-8 4 spaces Python 3.10
```

Flask 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('decision_model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/v_predict', methods=['POST'])
13 def v_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         preds="Worst performance with mileage " + str(prediction[0]) + "... Carry extra fuel"
25     if(output>9 and output<=17.5):
26         preds="Low performance with mileage " +str(prediction[0]) + "... Don't go to long distance"
27     if(output>17.5 and output<=29):
28         preds="Medium performance with mileage " +str(prediction[0]) + "... Go for a ride nearby."
29     if(output>29 and output<=46):
30         preds="High performance with mileage " +str(prediction[0]) + "... Go for a healthy ride"
31     if(output>46):
32         preds="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     """
```



```
21 print(prediction)
22 output=prediction[0]
23 if(output<9):
24     preds="Worst performance with mileage " + str(prediction[0]) + ". Carry extra fuel"
25 if(output>9 and output<=17.5):
26     preds="Low performance with mileage " +str(prediction[0]) + ". Don't go to long distance"
27 if(output>17.5 and output<=29):
28     preds="Medium performance with mileage " +str(prediction[0]) + ". Go for a ride nearby."
29 if(output>29 and output<=46):
30     preds="High performance with mileage " +str(prediction[0]) + ". Go for a healthy ride"
31 if(output>46):
32     preds="Very high performance with mileage " +str(prediction[0])+". You can plan for a Tour"
33
34 return render_template('index.html', prediction_text='{}'.format(preds))
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)
```

Integrate Flask Code :

```
PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
Machine learning based vehicle performance anal... Performance Prediction app.py
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3
4
5 import requests
6
7 API_KEY = "cPFRZC9gPSGv2pFnx5yDCBae0pgg8BLK8LdM8leln9v"
8 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": "urn:ibm:params:oauth:grant-type:apikey"})
9 mtoken = token_response.json()["access_token"]
10 headers = {'Content-Type': 'application/json', 'Authorization': 'Bearer '+mtoken}
11 app = Flask(__name__)
12
13 @app.route('/')
14 def home():
15     return render_template('index.html')
16
17 @app.route('/y_predict', methods=['POST', 'GET'])
18 def y_predict():
19     """
20     For rendering results on HTML GUI
21     """
22     x_test = [[int(x) for x in request.form.values()]]
23     print(x_test)
24     payload_scoring = {'input_data': [{"fields": ["f0", "f1", "f2", "f3", "f4", "f5"], "values": x_test}]}
25     response_scoring = requests.post('https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/0a65fea8-e525-4a26-987a-f668786b343e/predictions?version=2022-11-18', json=payload_scoring)
26     headers = {'Authorization': 'Bearer '+mtoken}
27     prediction = response_scoring.json()
28     output = prediction['predictions'][0]['values'][0][0]
29
30     if (output <= 9):
31         pred = "Worst performance with mileage " + str(output) + ". Carry extra fuel"
32     if (output > 9 and output <= 17.5):
33         pred = "Low performance with mileage " + str(output) + ". Don't go to long distance"
34     if (output > 17.5 and output <= 29):
35         pred = "Medium performance with mileage " + str(output) + ". Go for a ride nearby."
36     if (output > 29 and output <= 46):
37         pred = "High performance with mileage " + str(output) + ". Go for a healthy ride"
38     if (output > 46):
39         pred = "Very high performance with mileage " + str(output) + ". You can plan for a Tour"
40     return render_template('index.html', prediction_text=pred)
41
42 @app.route('/predict_api', methods=['POST'])
43 def predict_api():
44     """
45     For direct API calls through request
46     """
47     data = request.get_json(force=True)
48     prediction = model.y_predict([np.array(list(data.values()))])
49
50     output = prediction[0]
51     return jsonify(output)
52
53 if __name__ == "__main__":
54     app.run(debug=False)
```

```
Machine learning based vehicle performance anal... Performance Prediction app.py
25 payload_scoring = {'input_data': [{"fields": ["f0", "f1", "f2", "f3", "f4", "f5"], "values": x_test}]}
26 response_scoring = requests.post('https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/0a65fea8-e525-4a26-987a-f668786b343e/predictions?version=2022-11-18', json=payload_scoring)
27 headers = {'Authorization': 'Bearer '+mtoken}
28 prediction = response_scoring.json()
29 output = prediction['predictions'][0]['values'][0][0]
30
31 if (output <= 9):
32     pred = "Worst performance with mileage " + str(output) + ". Carry extra fuel"
33 if (output > 9 and output <= 17.5):
34     pred = "Low performance with mileage " + str(output) + ". Don't go to long distance"
35 if (output > 17.5 and output <= 29):
36     pred = "Medium performance with mileage " + str(output) + ". Go for a ride nearby."
37 if (output > 29 and output <= 46):
38     pred = "High performance with mileage " + str(output) + ". Go for a healthy ride"
39 if (output > 46):
40     pred = "Very high performance with mileage " + str(output) + ". You can plan for a Tour"
41 return render_template('index.html', prediction_text=pred)
42
43 @app.route('/predict_api', methods=['POST'])
44 def predict_api():
45     """
46     For direct API calls through request
47     """
48     data = request.get_json(force=True)
49     prediction = model.y_predict([np.array(list(data.values()))])
50
51     output = prediction[0]
52     return jsonify(output)
53
54 if __name__ == "__main__":
55     app.run(debug=False)
```

TESTING

TESTING REPORT

Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.

The Code was developed in 3 separate parts-

1. AI Model developed using Jupyter Notebook
2. Web Front end was developed using VS Code
3. Backend Database was developed using MongoDB

PROJECT NAME	MACHINE LEARNING VEHICLE PERFORMANCE ANALYZER
PROJECT TYPE	APPLIED DATA SCIENCE

DEVELOPER	KUMMITHA SIVA MOHAN REDDY
LANGUAGE	PYTHON,HTML,CSS,JAVA SCRIPT
TOTAL NUMBER OF TEST CASES	50
NUMBER OF TEST CASES EXECUTED	49
NUMBER OF TEST CASES PASSED	45
NUMBER OF TEST CASES FAILED	4-DUE TO TECHNICAL ISSUES

UNIT TESTING:

Unit testing is carried out screen-wise, each screen being identified as an object. Attention is diverted to individual modules, independently to one another to locate errors. This has enabled the detection of errors in coding and logic. This is the first level of testing. In this, codes are written such that from one module, we can move onto the next module according to the choice we enter.



SYSTEM TESTING:

In this, the entire system was tested as a whole with all forms, code, modules and class modules. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences.

It is a series of different tests that verifies that all system elements have been properly integrated and perform allocated functions.

System testing makes logical assumptions that if all parts of the system are correct, the goal will be successfully achieved. Testing is the process of executing the program with the intent of finding errors.

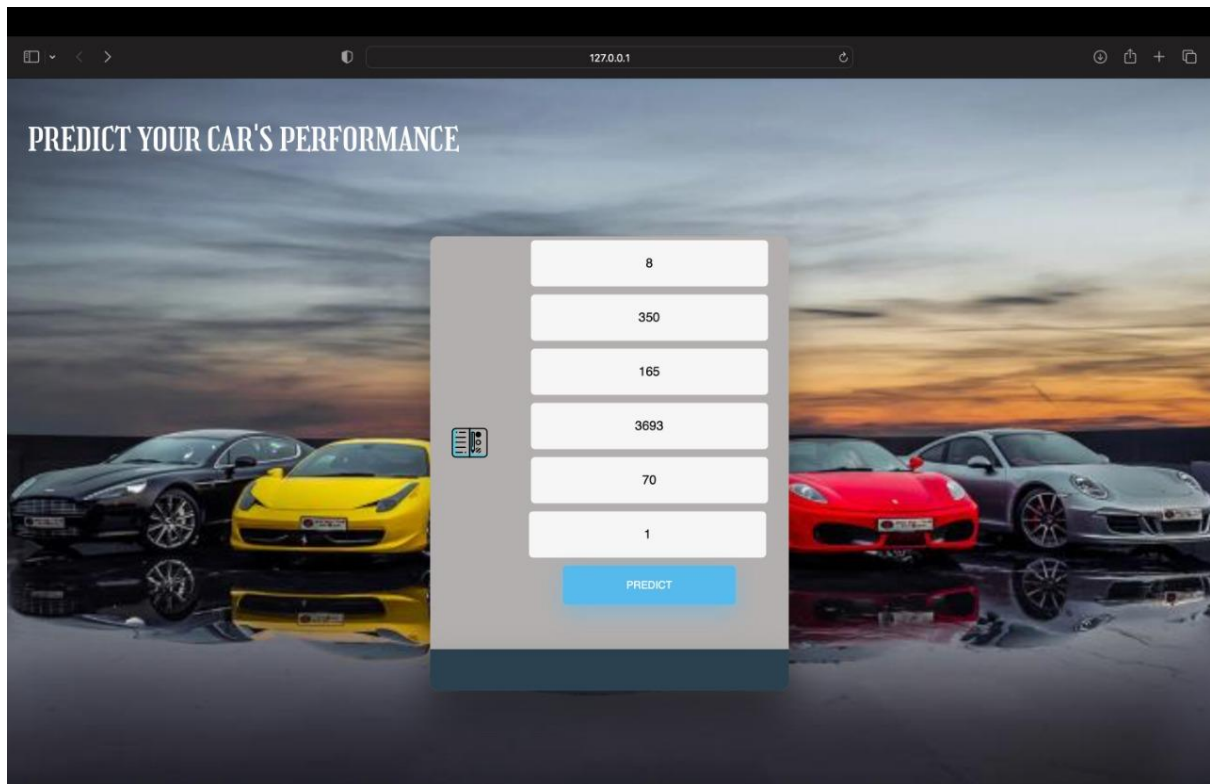
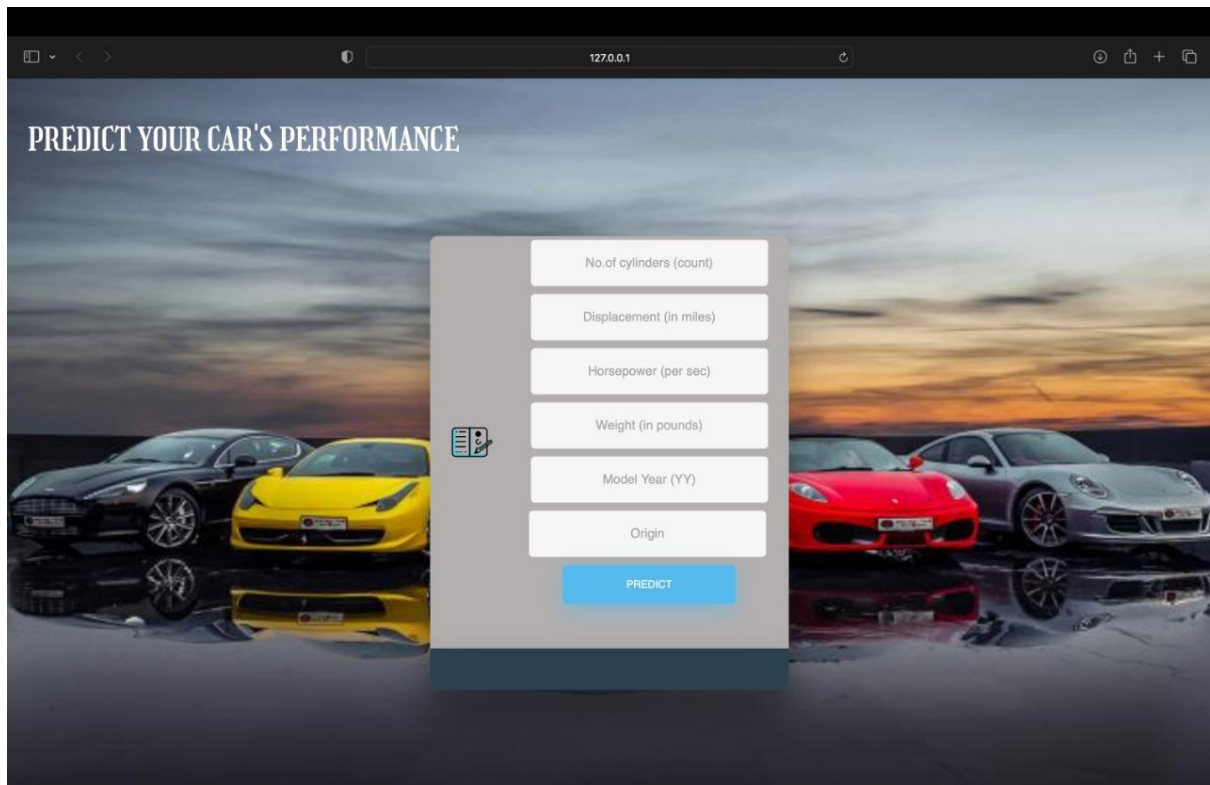
Testing cannot show the absence of defects, it can only show that software errors are present.

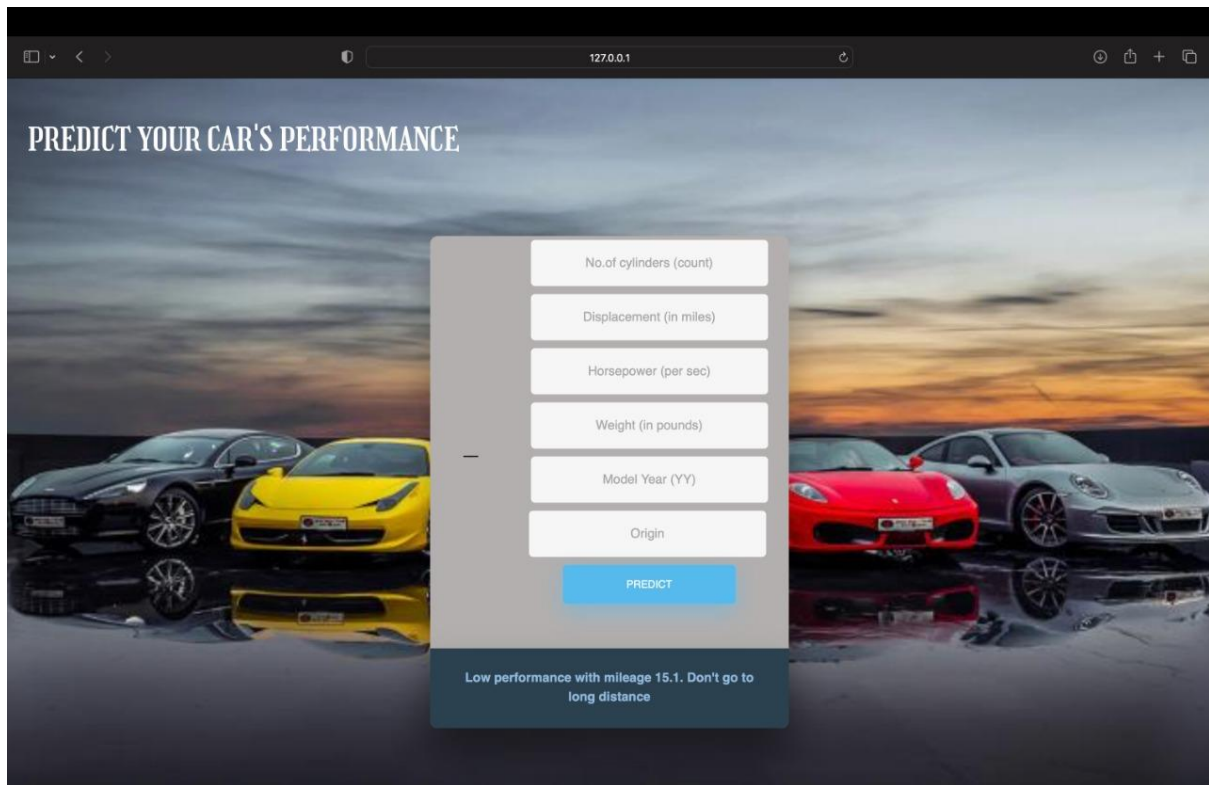


8.1 Test cases :

TEST CASE	No of Cylinders	Displacement	HP	Weight	Year	Origin	Predicted Value
1	8	307	130	3504	70	1	18.1
2	8	350	165	3693	70	1	15.2
3	4	130	95	2372	70	3	24.2
4	6	198	95	2833	70	1	22.3
5	4	104	95	2375	70	2	24.2

-





8.2 User Acceptance Testing :

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Machine Learning-based Vehicle Performance Analyzer project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	9	0	0	9
Client Application	44	0	0	44
Security	2	0	0	2

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

1. Test Case Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	15	6	2	3	26
Duplicate	1	0	3	0	4
External	2	3	0	1	6

Fixed	12	3	5	22	42
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	4	2	1	7
Totals	30	16	14	28	88

This report shows the number of test cases that have passed, failed, and untested

Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2










9 . Results :

9.1 Performance Metrics :

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
-------	-----------	--------	------------

1.	Metrics	Regression Model: MAE - 1.7841, MSE - 6.5057, RMSE -2.5506, R2 score – 0.9058	<div>  <pre>[63] #importing necessary libraries to find evaluation o from sklearn.metrics import mean_absolute_error as from sklearn.metrics import r2_score from sklearn.metrics import mean_squared_error import math</pre> </div> <div>   <pre># Mean Absolute Error MAE = mae(y_test, y_pred) print("MAE:",MAE)</pre> <pre>MAE: 1.7841771356783922</pre> </div> <div>  <pre>[65] #mean squared error MSE=mean_squared_error(y_test,y_pred) print("MSE:",MSE)</pre> <pre>MSE: 6.505788848703318</pre> </div> <div>  <pre>[66] #Root mean squared error RMSE=math.sqrt(MSE) print("RMSE:",RMSE)</pre> <pre>RMSE: 2.550644790774152</pre> </div> <div>  <pre>[67] #checking the performance of the model using r2_sco r2=r2_score(y_test,y_pred) print("R2_score:",r2)</pre> <pre>R2_score: 0.9058760463516443</pre> </div>
2.	Tune the Model	Hyperparameter Tuning –	<div>  <pre>[41] from sklearn.ensemble import RandomForestRegres</pre> </div> <div>   <pre>rf= RandomForestRegressor(n_estimators=10,randomo model=rf.fit(x_train,y_train)</pre> </div>

TEST CASE	No of Cylinders	Displacement	HP	Weight	Year	Origin	Predicted Value
1	4	120	97	2506	72	3	23
2	4	98	80	2164	72	1	28
3	4	97	88	2100	72	3	27
4	8	350	175	4100	73	1	13
5	8	304	150	3672	73	1	14

10. ADVANTAGES & DISADVANTAGES :

ADVANTAGES:

- It helps users for predicting the vehicle performance.
- Here the chance of occurrence of error is less when compared with the existing system.
- It is fast, efficient and reliable.
- Avoids data redundancy and inconsistency.
- Very user-friendly.
- Easy accessibility of data

DISADVANTAGES:

- computer literacy and network access
- Low Computer Literacy
- Security Concerns
- Authenticity

Infrastructural Requirement.

10 . Conclusion :

The monitoring of car performance, especially gas consumption, has so far been approached only very superficially. A typical fuel gauge, when closely monitored, shows an extremely non-linear relationship between needle movement and fuel consumption. Inaccuracies occur especially in the range of critical low fuel values of 5-10% or more. In the past, due to this limitation, some luxury cars had an audible and flashing light alarm function to indicate a

low fuel condition. These systems, which add to the existing fuel level, have no more accuracy than the fuel level monitor alone. In recent years, with the availability of computer techniques and reliable and less expensive computer equipment, a number of systems have been developed to provide somewhat more accurate information about vehicle performance.

12. FUTURE SCOPE :

This merits exploratory methods based on actual failures to deduct likely failure modes. This thesis presents two methods for data mining the vehicle maintenance records and vehicle usage data to learn usage or wear patterns indicative of failures. This requires detailed maintenance records where the failure root cause can be deducted with accurate date or mileages of the repair.

Further, more wide-spread adoption of predictive maintenance calls for automatic and less human-resource demanding methods, e.g. unsupervised algorithms with lifelong learning. Such methods are easier to scale up and they can thus be ubiquitously applied since much of the modelling is automated and requires little or no human interaction.

Maintenance predictions can be enhanced by combining the deviations in onboard data with off-board data sources such as maintenance records and failure statistics. This is exclusive product knowledge, only accessible to the vehicle manufacturers, which gives them an advantage in predicting maintenance. Still, data mining has yet to become a core competence of vehicle manufacturers, which makes the road to industrialisation long.

The aim of this thesis is to investigate how on-board data streams and off-board data can be used to predict the vehicle maintenance. More specifically, how on-board data streams can be represented and compressed into a transmittable size and still be relevant for maintenance predictions. Further, the most interesting deviations must be found for a given repair which requires new ways of combining semantic maintenance records with deviations based on compressed on-board data.

This can be accessed anytime anywhere, since it is a web application provided only an internet connection.

13. APPENDIX :

Github Link : - <https://github.com/IBM-EPBL/IBM-Project-21845-1659792766>