Machine Learning based Vehicle Performance Analyzer

Technology: Applied Data Science

IBM-Project-23359-1659880124

Team ID: PNT2022TMID02351

A PROJECT REPORT

Submitted by

KAVINKUMAR. G	**(190801082)**
HARIPRASAD. R	**(190801504)**
HAREESH. S	**(190801055)**
KARTHIKEYAN. A	**(190801080)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS)

CHENNAI - 602105

CHAPTER 1
INTRODUCTION

1.1PROJECT OVERVIEW

Predicting a car's performance is a significant and intriguing challenge. The current study's main goal is to forecast automobile performance in order to improve specific vehicle behaviour. This can significantly reduce the system's fuel consumption and increase its effectiveness. Analysis of vehicle performance based on engine type, cylinder count, fuel type, and horsepower, among other factors. These variables can be used to forecast the health of the vehicle. It is a continuous process to collect, investigate, interpret, and document health data based on the three elements. Both prediction engines and engine management systems place a high value on performance metrics such as mileage, reliability, flexibility, and cost, which can be combined. To improve the vehicle's performance efficiency, it is critical to analyse the elements using a variety of well-known machine learning methodologies, such as linear regression, decision trees, and random forests. The power, lifespan, and range of automotive traction batteries are currently "hot topics" in automotive engineering. In this case, we also consider mileage performance. To solve this problem, we will build models using various techniques and neural networks. Then, we'll see which algorithm best predicts car performance (Mileage).

1.2 PURPOSE

The application of Machine Learning (supervised and unsupervised) techniques to automotive engine sensor data in order to discover driver usage patterns and perform classification via a distributed online sensing platform. These platforms can be used in a variety of domains, including fleet management, the insurance market, fuel consumption optimization, and CO2 emission reduction. Thus, the project's main goal is to predict the performance of the car in order to improve certain vehicle behaviors using various machine learning algorithms.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING PROBLEM

Since the development of new technologies, the potential for processing car sensing data has increased in recent years. This type of data is useful for analyzing how drivers behave behind the wheel, for example. Very little has been done to analyze car usage patterns based on car engine sensor data, and thus it has not been explored to its full potential by taking into account all sensors within a car engine. To bridge this gap, the use of Machine Learning techniques (supervised and unsupervised) on automotive engine sensor data to discover drivers' usage patterns, Such platforms can be used in a variety of domains, including fleet management, insurance markets, fuel consumption optimization, and CO2 emission reduction, among others

2.2 PROBLEM DEFINITION

As a result of going through the existing problem and learning from the various papers in the literature survey. The problem definition can be framed as follows:
"To predict the performance of the car in order to improve certain vehicle behaviors using various machine learning algorithms.

2.3 REFERENCE

2.3.1 ML Based Real-Time Vehicle Data Analysis for Safe Driving Modeling

In the paper “Machine Learning Based Real-Time Vehicle Data Analysis for Safe Driving Modeling” Machine learning approach to analyze and predict the vehicle performance in real time. The focus is on analyzing the data which is collected from the vehicle using the OBD-II scanner and eventually providing the driver’s safety solutions The meta features of the vehicle are analyzed in the cloud and are then shared to the concerned parties. The proposed system consists of an OBD-II scanner and a mini dash cam which continuously send data to the cloud server where data analysis is done.

Multivariate Linear Regression Model:

It is used when we want to predict the value of a variable based on the value of two or more different variables. The variable we want to predict is called the Dependent Variable, while those used to calculate the dependent variable are termed as Independent Variables.

Features such as fuel efficiency, average speed value, maximum speed value, fourth section speed value, interval driving distance, driving time value during green zone, traveling time value, emergency accelerated value, emergency decelerated value, fourth rpm time value and fifth rpm time value are used for training the model.

The real time data obtained is normalized using Min-Max normalization technique and they hypothesize an outcome called Economic Driving Index (ECN_DRVG_INDX) and another outcome called Safe Driving Index (SFTY_DRVG_INDX). The results have proven to be approximately 80% fitting the given features.

2.3.2 Machine Learning Approach Based on Automotive Engine Data Clustering for Driver Usage Profiling Classification:

The paper “A Machine Learning Approach Based on Automotive Engine Data Clustering for Driver Usage Profiling Classification” proposes the use of Machine Learning techniques (supervised and unsupervised) on automotive engine sensor data to discover drivers’ usage patterns, and to perform classification through a distributed online sensing platform and that such platform can be useful used in different domains, such as fleet management, insurance market, fuel consumption optimization, CO2 emission reduction, among others.

As automotive engine data has no class label, we use the following Machine Learning models used for clustering and class labels:

K means:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Expectation-Maximization:

The expectation-maximization algorithm is an approach for performing maximum likelihood estimation in the presence of latent variables. It does this by first estimating the values for the latent variables, then optimizing the model, then repeating these two steps until convergence. It is an effective and general approach and is most used for density estimation with missing data, such as clustering algorithms like the Gaussian Mixture Model.

Hierarchical Clustering:

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster. In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.

Machine learning algorithms for Classification:

Decision Tree:

The decision tree and its variants are the other learning algorithms that divide the input space into regions and has separate parameters for each region. They are classified as a non- parametric supervised learning method which is widely used in classification and regression, as well as in representing decisions and decision making. The structure of a decision tree is a flowchart, in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Besides, the paths from root to leaf represent classification rules

KNN:

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

Multilayer Perceptron:

A multilayer perceptron is a fully connected class of feedforward artificial neural network. it uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Random Forest

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training setRandom forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Support Vector Mechanism:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

2.3.3 Driving Behavior Analysis Using Machine and Deep Learning Methods for Continuous Streams of Vehicular Data:

The paper “Driving Behavior Analysis Using Machine and Deep Learning Methods for Continuous Streams of Vehicular Data” authored by Nikolaos Peppes, Theodoros Alexakis, Evgenia Adamopoulou, Konstantinos Demestichas aims to combine well-known machine and deep learning algorithms together with open-source-based tools to gather, store, process, analyze and correlate different data flows originating from vehicles

Machine Learning Algorithms for Classification:

Support Vector Mechanisms (SVM):

Support vector machines is a supervised machine learning algorithm used for both classification and regression. SVM classifies data points based on the hyperplane in an N – dimensional space. The separation function in support vector classification is a linear combination of kernels linked to the support vector.

Decision Tree-Based Algorithms:

The decision tree and its variants are the other learning algorithms that divide the input space into regions and have separate parameters for each region. They are classified as a nonparametric supervised learning method which is widely used in classification and regression, as well as in representing decisions and decision making. The structure of a decision tree is a treelike flowchart, in which each internal node represents a “test” on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Besides, the paths from root to leaf represent classification rules. Three decision tree-based models, including decision tree (DT), extra trees (ExT), and random forest, were evaluated in relation to various learning method

Random Forest

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Deep Learning Model:

RNN-based algorithms:

RNN-based models have been used widely nowadays due to its robustness and capability to handle nonlinear data even with its typically structured, single hidden layer, or advanced structured, multiple hidden layers. RNN includes three layers: input, hidden, and output layers. In case of increasing complexity of the problem, the number of layers will rise, and the computational resources will consequently also rise. Here, both the mentioned structures of the RNN-based models were utilized for predicting the Driving Behavioral Analysis.

Multilayer Perceptron:

A multilayer perceptron is a fully connected class of feedforward artificial neural network. it uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

CHAPTER 3
IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP

The primary goal of the empathy map is to bridge the gap between the user and the developer. The empathy map for the machine learning-based vehicle performance analyzer is represented in Fig 3.1.

Figure 3.1 – Empathy Map

3.2 IDEATION & BRAINSTORMING

This is quite often the most exciting stage of a project because the goal of Ideation and brainstorming is to generate a large number of ideas that the team can then filter and cut down into the best, most practical, or most innovative ones to inspire new and better design solutions and products. The stages of ideation and brainstorming for the machine learning-based vehicle performance analyzer are shown in Figure 3.2.

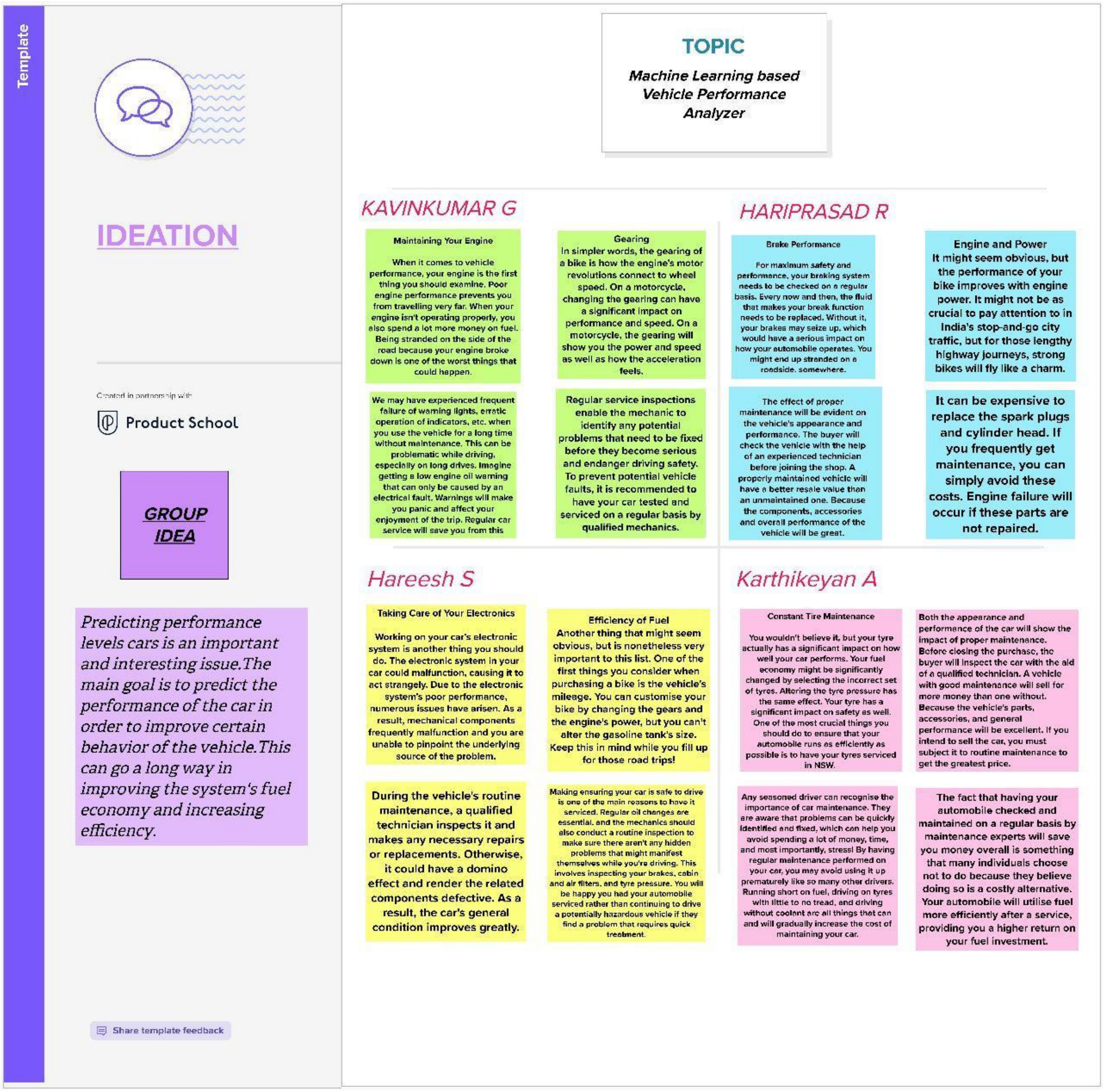


Figure 3.2 – Ideation & Brainstorming

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In our day-to-day life vehicles play an integral part in all our lives. We are lagging in servicing it because we are preoccupied with our daily tasks which have a direct impact on the environment. To improve the performance and efficiency of our vehicle, it must be properly serviced and maintained. So, to remind the owner about vehicle maintenance, we have created a Vehicle Performance Analyzer using machine learning.
2.	Idea / Solution description	To improve the vehicle's performance efficiency, it is critical to analyze the factors using a variety of well-known machine learning algorithms such as linear regression, decision tree, and random forest. Automotive traction batteries' range, durability, and longevity are "hot topics" in automotive engineering. And now we'll look at mileage performance. We will create models using various algorithms and neural networks to solve this problem. We'll then see which algorithm accurately predicts car performance and its efficiency. This can significantly reduce system fuel consumption and increase efficiency.
3.	Novelty / Uniqueness	There are a few works that analyze vehicle performance using very few vehicle parameters, whereas, in our idea, we use the number of cylinders, displacement, horsepower, weight, model’s year and country of origin to determine vehicle performance. We anticipate that as more data is added to fit the model, the sensitivity of our measure will increase. Because our model will be exposed to more possible scenarios, it will be able to find more data that is similar to the previously unseen ones.

4.	Social Impact / Customer Satisfaction	The main objective of this Vehicle performance analyser is that it helps in major reduction of emissions from the vehicles. The reduced amount of poisonous gas emission will definitely improve the quality of air in our environment. By using this Vehicle Performance Analyser customers can know the technical status of their own vehicle. It allows the customer to maintain good quality of the vehicle by enhancing the engine performance, taking care of the interior, regular tire maintenance and also improves the driver safety whereas vehicle gives service alerts which provides better driving experience.
5.	Business Model (Revenue Model)	This System will provide detailed information about the vehicle performance and very user- friendly interface to use. By being informative and unique, it attracts more customers leading to higher revenue. As it plays a vital role in maintaining the efficiency of the vehicle and also in saving the environment from global warming it has a greater impact on the competitive business world.
6.	Scalability of the Solution	Irrespective of the vehicle type or the count of vehicles, this system will analyse the performance of the vehicle and also gives periodic service alerts, when performance of the vehicle degrades. Multiple users can also access the system at same time, it processes the results without any delay.

3.4 PROBLEM SOLUTION FIT

The problem solution fit is the solution found to address the customer's problem.

The solution for the machine learning-based vehicle performance analyzer is depicted in Figure 3.4.

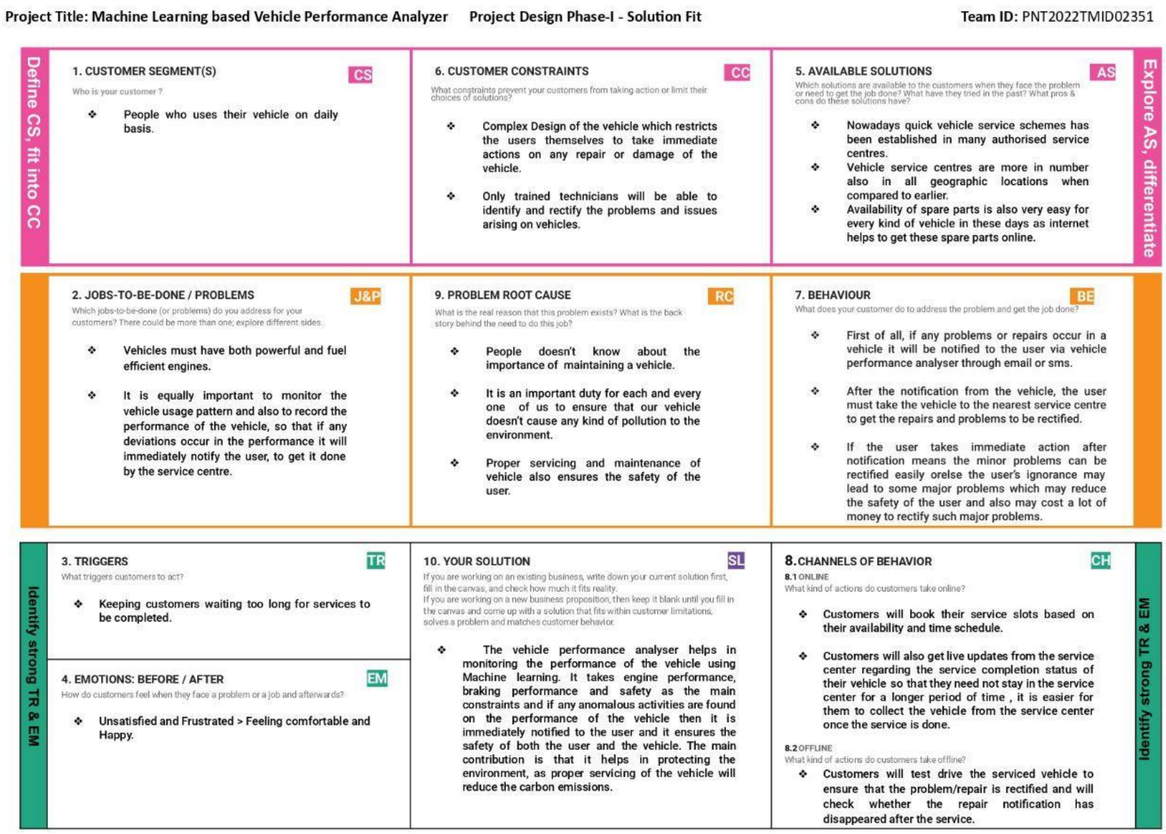


Figure 3.4 – Problem Solution Fit

CHAPTER 4
REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Data Collection	Form-based user input sending the data to the server
FR-4	Query Processing Predict	The ML model to forecast the anticipated mileage. Search for more recent vehicles that resemble the current model.
FR-5	Report Generation	Display the anticipated distance and plot the anticipated mileage over time. suggest related auto models based on the database.

4.2 NON-FUNCTIONAL REQUIREMENTS

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To collect data, this programme doesn't need any new, specialized hardware. Through data that the user can manually gather, it attempts to estimate mileage.
NFR-2	**Security**	Ensured Confidentiality, Integrity, and Availability while being protected from all types of web-based threats.
NFR-3	**Reliability**	In terms of the efficiency and remaining life of the car, the programme will provide nearly perfect predictions, and it will be designed in such a way that false positives won't negatively impact users in anyway.
NFR-4	**Performance**	The performance of this application can handle a sizable number of concurrent users accessing the services with little to no apparent impact.
NFR-5	**Availability**	Minimizing service downtime by ensuring that the the programme is always accessible to all users.
NFR-6	**Scalability**	All types of vehicles, not just cars, could use this application.

CHAPTER 5
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a classic visual representation of how data flows within a system. A neat and clear DFD can thus graphically depict the appropriate amount of system requirements. It demonstrates not only how data enters and exits the system, but also what changes the information and where it is stored. The DFD for the given project is depicted in Figure 5.1.

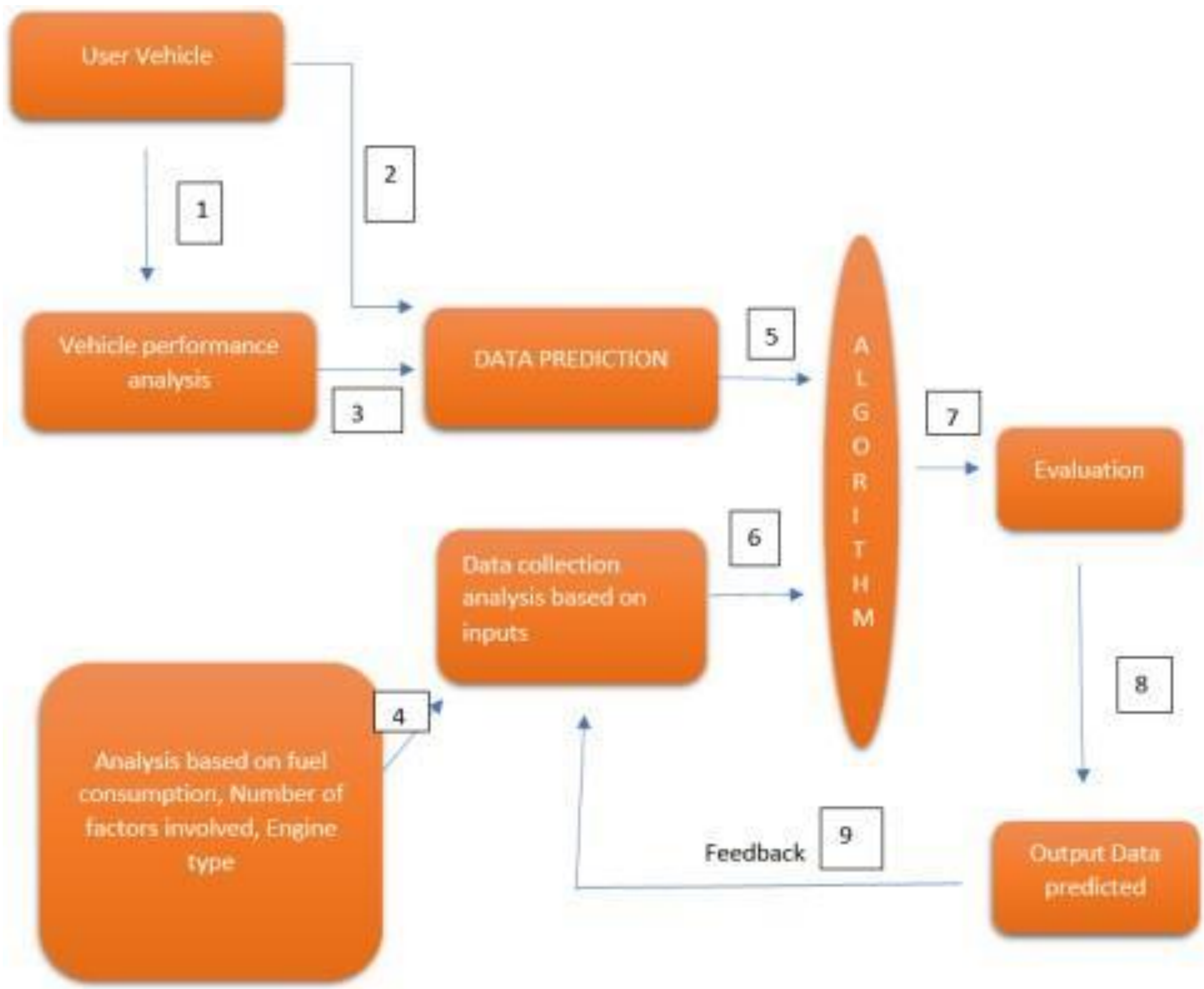


Figure 5.1 – Data Flow Diagram

5.2 SOLUTION & TECHNICAL ARCHITECTURE

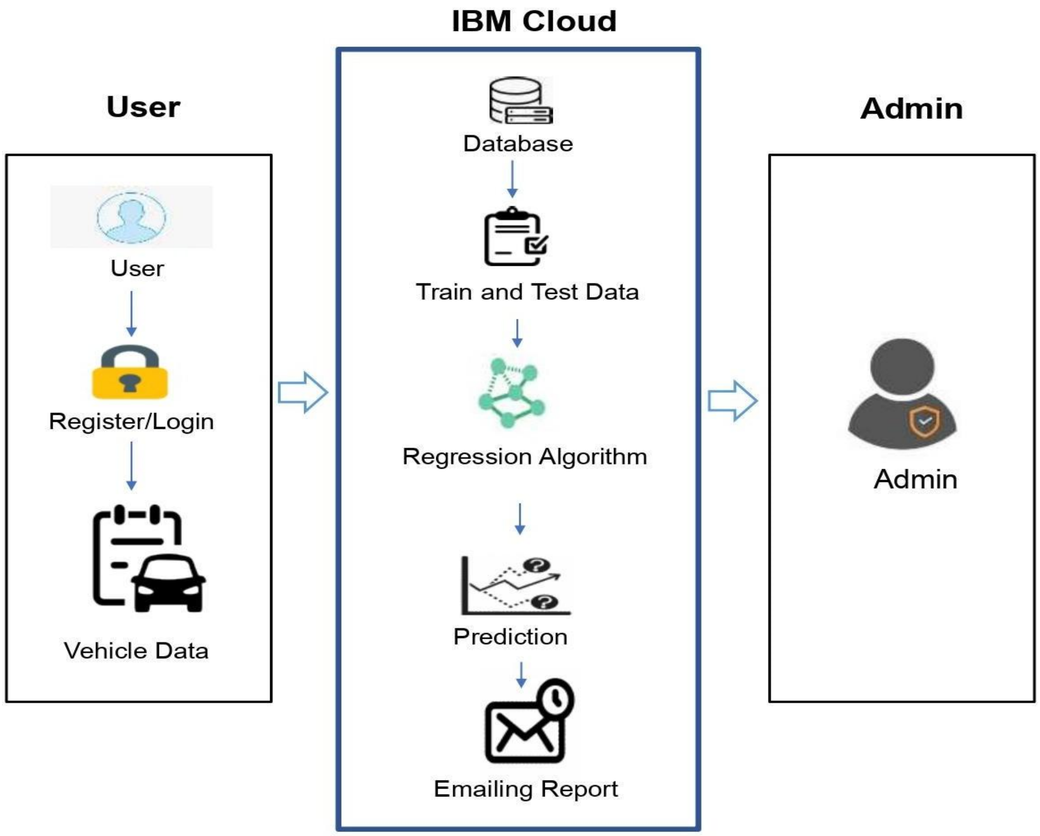


Figure 5.2 Technical Architecture

5.2.1 COMPONENTS AND TECHNOLOGIES

S.No	Component	Description	Technology
1.	User Interface	The user interacts with the application through a Web Application that is responsive to the device that is being used.	React Js
2.	Get User Data	The process collects the user input data that is collected via a form to the server as a JSON Object	REST API
3.	Model Prediction	Use the data collected from the user to make predictions on the mileage expected.	IBM Watson ML
4.	Send User Report	Send the predictions along with suggestions to the user as JSON Object	REST API
5.	Database	Databases contain user information such as name, email, vehicle basic information, mileage predicted over time.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	External API-1	Vehicle Details Database	https://api.auto-data.net/
8.	Machine Learning Model	The machine learning model is used to predict mileage from the user inputs	Regression Modelling.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Core i5, 8GB RAM Cloud Server Configuration:	Local, Docker

Table 5.2.1 – Components and Technologies

5.2.2 APPLICATION CHARACTERISTICS

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	React Js, Flask, Sci-kit Learn	Javascript, Python
2	Security Implementations	Identity and Access Management, OAUTH, WAF	IBM Cloud
3	Scalable Architecture	3 Tier Architecture, Model-View-Controller implementation	Model - SQL DB, View - ReactJS, Controller - Flask Server
4	Availability	Proxy servers, Load Balancers to help balance traffic among servers to help improve uptime	IBM Cloud load balancers
5	Performance	The frontend is detached from the Business logic server reduces requests sent to the server.	Nginx proxy

Table 5.2.2 – Application Characters

5.3 USER STORIES

User Type	Functional Requirements	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer	Access the Webpage	USN -1	Anyone can access the webpage to check the specifications of the vehicle	I can access my webpage online at any time	High	Sprint-1
Customer	Performance of the Vehicle	USN - 2	As per the usage of the user, the performance of the vehicle should be predictable.	Prediction can be done in an easy way.	High	Sprint-2
Customer	Accuracy to check the performance and health of the car	USN -3	By using our prediction, it helps to check the health of the car.	The efficiency of the car can be predicted.	High	Sprint-3

Table 5.3 – User Stories

CHAPTER 6
PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data processing	USN-1	As a user, I can process raw data and perform manual analysis.	30	High	Kavinkumar G Karthikeyan A Hareesh S Hari prasad R
Sprint-2	Model building	USN-2	As a user, I can get the predicted performance of the vehicle using the given data.	20	Low	Kavinkumar G Karthikeyan A Hareesh S Hari prasad R
Sprint-3	Web Page design	USN-3	As a user, I am able to view the website and I can get the predicted performance of the vehicle using the given data.	30	High	Kavinkumar G Karthikeyan A Hareesh S Hari prasad R
Sprint-4	Result	USN-4	As a user, I expect the prediction is highly accurate.	20	High	Kavinkumar G Karthikeyan A Hareesh S Hari prasad R

Table 6.1 – Sprint Planning

6.2 SPRINT DELIVERY SCHEDULE

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	30	1 Day	01 Nov 2022	03 Nov 2022	30	12 Nov 2022
Sprint-2	20	2 Days	03 Nov 2022	05 Nov 2022	20	12 Nov 2022
Sprint-3	20	5 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	4 Days	12 Nov 2022	16 Nov 2022	20	16 Nov 2022

Table 6.2 – Sprint Delivery Schedule

6.3 REPORT FOR JIRA

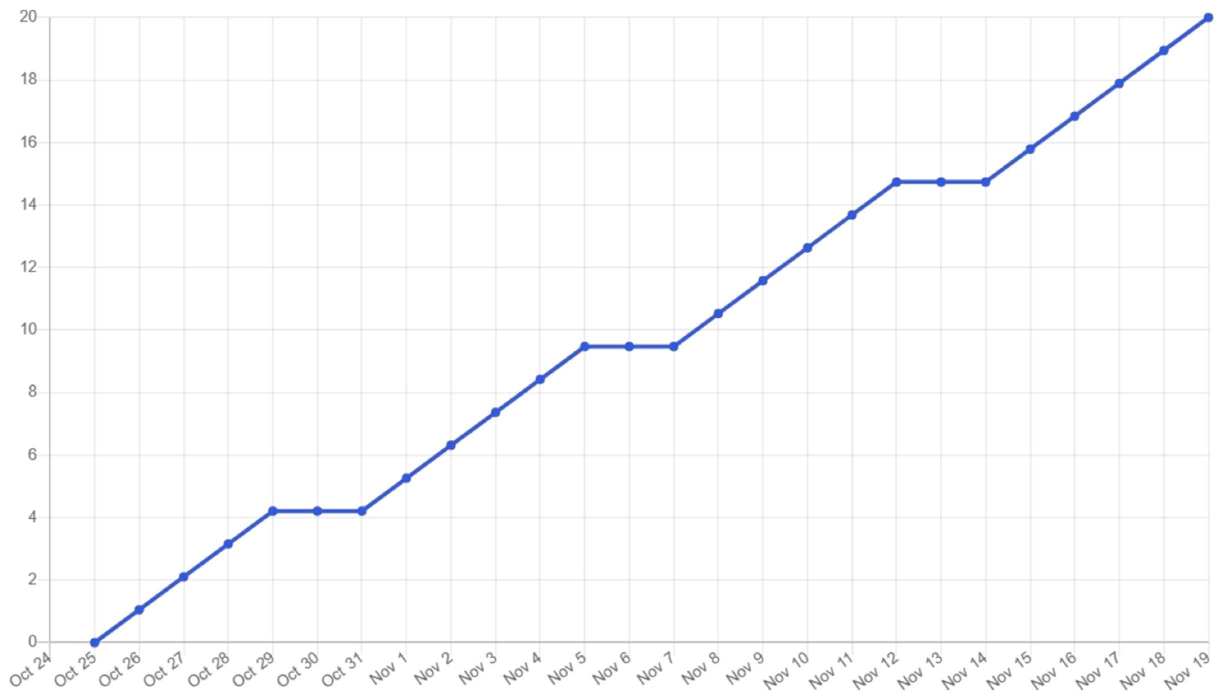
Velocity :

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)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



CHAPTER 7
CODING AND SOLUTION

Feature 1:
Random Forest Regressor

Random Forest Regressor

In [49]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [50]:
```
x11 = dataset.iloc[:,1:8].values
y11 = dataset.iloc[:,0].values
```

In [51]:
```
from sklearn.model_selection import train_test_split
x_train1, x_test1, y_train1, y_test1 = train_test_split(x11,y11,test_size=0.2,random_state=0)
```

In [52]:
```
rf= RandomForestRegressor(n_estimators=30,random_state=0)
rf.fit(x_train1,y_train1)
```

Out[52]:
```
RandomForestRegressor(n_estimators=30, random_state=0)
```

In [53]:
```
y1_pred=rf.predict(x_test1)
y1_pred
```

Out[53]:
```
array([14.3       , 24.34333333, 14.18333333, 20.26666667, 18.43333333,
       30.21666667, 34.96      , 21.3       , 15.36666667, 26.22333333,
       36.01333333, 36.5       , 18.95666667, 27.22333333, 16.47666667,
       32.54333333, 27.89333333, 27.17      , 16.86666667, 34.64333333,
       15.88333333, 23.3       , 23.48333333, 20.71666667, 32.22      ,
       27.23333333, 34.40666667, 30.03      , 31.76333333, 15.93333333,
       19.07666667, 33.32333333, 18.55      , 32.66      , 20.35666667,
       24.2       , 18.92      , 16.40666667, 35.24      , 12.3       ,
       13.4       , 15.4       , 27.89666667, 32.61333333, 29.06666667,
       22.1       , 19.83      , 14.8       , 22.11333333, 29.86666667,
       34.04      , 25.36666667, 16.34      , 27.4       , 15.4       ,
       12.36666667, 18.56666667, 25.32666667, 31.78333333, 16.24      ,
       18.87      , 25.77666667, 18.96666667, 21.53333333, 13.26666667,
       15.11666667, 13.46666667, 17.26333333, 24.95666667, 14.       ,
       35.61333333, 13.3       , 23.01333333, 18.2       , 23.90333333,
       29.51666667, 27.1       , 30.97      , 29.67666667, 14.35      ])
```

Feature 2:
Accuracy

In [54]:

```
from sklearn.metrics import r2_score
accuracy = r2_score(y_test1, y1_pred)
accuracy
```

Out[54]: 0.8999792555413947

Dataset:

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11	70	1	plymouth satellite
16	8	304	150	3433	12	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
15	8	429	198	4341	10	70	1	ford galaxie 500
14	8	454	220	4354	9	70	1	chevrolet impala
14	8	440	215	4312	8.5	70	1	plymouth fury iii
14	8	455	225	4425	10	70	1	pontiac catalina
15	8	390	190	3850	8.5	70	1	amc ambassador dpl
15	8	383	170	3563	10	70	1	dodge challenger se
14	8	340	160	3609	8	70	1	plymouth 'cuda 340
15	8	400	150	3761	9.5	70	1	chevrolet monte carlo
14	8	455	225	3086	10	70	1	buick estate wagon (sw)
24	4	113	95	2372	15	70	3	toyota corona mark ii
22	6	198	95	2833	15.5	70	1	plymouth duster
18	6	199	97	2774	15.5	70	1	amc hornet
21	6	200	85	2587	16	70	1	ford maverick
27	4	97	88	2130	14.5	70	3	datsun pl510
26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan
25	4	110	87	2672	17.5	70	2	peugeot 504
24	4	107	90	2430	14.5	70	2	audi 100 ls
25	4	104	95	2375	17.5	70	2	saab 99e
26	4	121	113	2234	12.5	70	2	bmw 2002
21	6	199	90	2648	15	70	1	amc gremlin
10	8	360	215	4615	14	70	1	ford f250

CHAPTER 8
TESTING

8.1 TEST CASE

Test case ID	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Home Page	Webpage is opened after clicking the given URL.		1.Enter URL and click enter.	http://127.0.0.1:5000/	It should display the Vehicle performance analyser webpage.	Working as expected	Pass	Obtained the expected results.			KAVINKUMAR G HARIPRASAD R HAREESH S KARTHIKEYAN A
LoginPage_TC_OO2	Home Page	Verify the UI elements in Login/Signup popup		1.Enter the values for the requirements as per the vehicle details.	http://127.0.0.1:5000/y_	It should display the performance status of vehicle along with milaege and a comment.	Working as expected	Pass	Obtained the expected results.			KAVINKUMAR G HARIPRASAD R HAREESH S KARTHIKEYAN A

8.2 USER ACCEPTANCE TESTING

3. Test Case Analysis

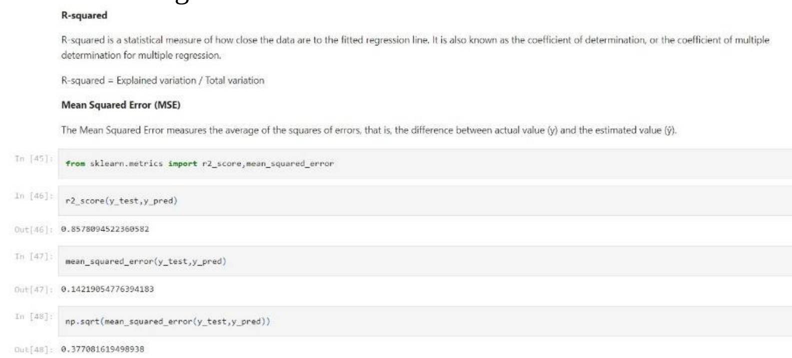
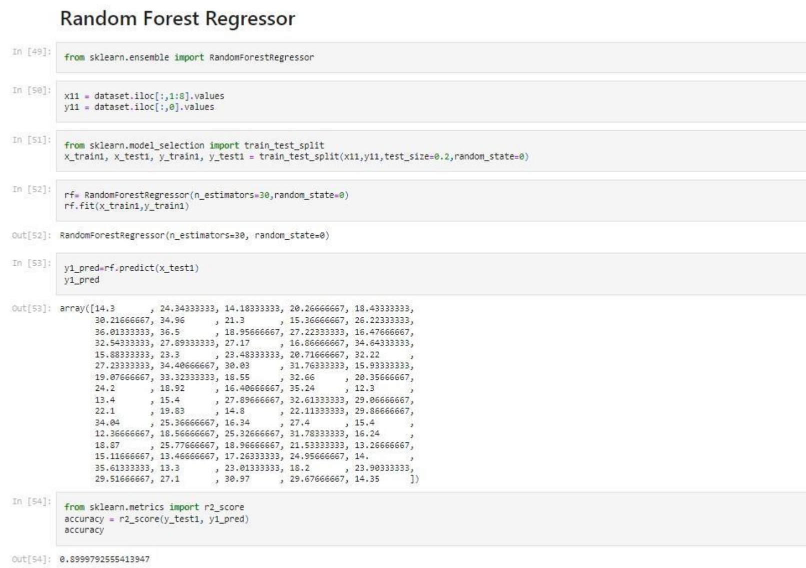
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
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

CHAPTER 9
RESULTS

9.1 PERFORMANCE METRICS

<table>
<tr><th>S. No.</th><th>PARAMETER</th><th>VALUES</th><th>SCREENSHOT</th></tr>
<tr><td>1.</td><td>Metrics</td><td>Regression Model:
MAE-,MSE-,RMSE-,R2 score-

Classification Model:
Confusion Matrix, Accuray Score- & Classification Report-</td><td>Decision tree regression

R-squared
R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.
R-squared = Explained variation / Total variation
Mean Squared Error (MSE)
The Mean Squared Error measures the average of the squares of errors, that is, the difference between actual value (y) and the estimated value (ŷ).
In [45]: from sklearn.metrics import r2_score,mean_squared_error
In [46]: r2_score(y_test,y_pred)
Out[46]: 0.8578094522360582
In [47]: mean_squared_error(y_test,y_pred)
Out[47]: 0.14219054776394183
In [48]: np.sqrt(mean_squared_error(y_test,y_pred))
Out[48]: 0.377081619498938</td></tr>
<tr><td></td><td></td><td></td><td>
Random Forest Regressor
In [49]: from sklearn.ensemble import RandomForestRegressor
In [50]: x11 = dataset.iloc[:,1:8].values
y11 = dataset.iloc[:,0].values
In [51]: from sklearn.model_selection import train_test_split
x_train1, x_test1, y_train1, y_test1 = train_test_split(x11,y11,test_size=0.2,random_state=0)
In [52]: rf= RandomForestRegressor(n_estimators=30,random_state=0)
rf.fit(x_train1,y_train1)
Out[52]: RandomForestRegressor(n_estimators=30, random_state=0)
In [53]: y1_pred=rf.predict(x_test1)
y1_pred
Out[53]: array([14.3 , 24.34333333, 14.18333333, 20.26666667, 18.43333333,
30.21666667, 34.96 , 21.3 , 15.36666667, 26.22333333,
36.01333333, 36.5 , 18.95666667, 27.22333333, 16.47666667,
32.54333333, 27.89333333, 27.17 , 16.86666667, 34.64333333,
15.88333333, 23.3 , 23.48333333, 20.71666667, 32.22 ,
27.23333333, 34.40666667, 30.03 , 31.76333333, 15.93333333,
19.07666667, 33.32333333, 18.55 , 32.66 , 20.35666667,
24.2 , 18.92 , 16.40666667, 35.24 , 12.3 ,
13.4 , 15.4 , 27.89666667, 32.61333333, 29.06666667,
22.1 , 19.83 , 14.8 , 22.11333333, 29.86666667,
34.04 , 25.36666667, 16.34 , 27.4 , 15.4 ,
12.36666667, 18.56666667, 25.32666667, 31.78333333, 16.24 ,
18.87 , 25.77666667, 18.96666667, 21.53333333, 13.26666667,
15.11666667, 13.46666667, 17.26333333, 24.95666667, 14. ,
35.61333333, 13.3 , 23.01333333, 18.2 , 23.90333333,
29.51666667, 27.1 , 30.97 , 29.67666667, 14.35])
In [54]: from sklearn.metrics import r2_score
accuracy = r2_score(y_test1, y1_pred)
accuracy
Out[54]: 0.8999792555413947

Mean Squared error</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<pre>
In [60]: from sklearn.metrics import r2_score,mean_squared_error

In [61]: r2_score(y_test,y_pred2)

Out[61]: -0.04347826086956519

In [62]: mean_squared_error(y_test,y_pred2)

Out[62]: 0.6

In [63]: np.sqrt(mean_squared_error(y_test,y_pred2))

Out[63]: 0.7745966692414834
</pre>
Linear regression
<pre>
In [44]: ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred, 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()

C:\Users\Max_l\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\Max_l\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
</pre>


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.
</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>Training accuracy-0.8999792555413947</td>
<td>
<pre>
In [54]: from sklearn.metrics import r2_score
accuracy = r2_score(y_test1, y1_pred)
accuracy

Out[54]: 0.8999792555413947
</pre>
</td>
</tr>
</table>

Figure 9.1 – Performance Metrics

CHAPTER 10
PROS AND CONS

Pros

• Using the Random Forest Algorithm in the model aids in classification and regression tasks.
• A random forest produces good predictions that are easy to understand
• It can easily handle large datasets
• The Random Forest Algorithm predicts outcomes with a higher level of accuracy.

Cons

• The main limitation of using random forest algorithm in the model is that a large number trees can make the algorithm too slow and ineffective for real-time predictions.
• The random forest algorithm is quite slow to create predictions once it is trained.

CHAPTER 11
CONCLUSION

Estimating a car's performance level is a significant and fascinating challenge. Our main goal was to forecast vehicle performance so that we could improve specific vehicle behavior. The car's performance is assessed based on factors such as horsepower, cylinder count, fuel type, and engine type, among others. The health of the car is forecasted based on factors such as horsepower, cylinder count, fuel type, and engine type. To optimize the vehicle's performance efficiency, we analyzed the components using a variety of well-known machine learning approaches such as linear regression, decision trees, and random forests. The power, longevity, and range of automobile traction batteries have recently become "hot topics" in automotive engineering. In this case, we also take mileage performance into account. We built the models to solve this problem using a variety of methods and neural networks. We then compared which algorithm is best at forecasting car performance (Mileage). A front-end web page was created to assist the user in presenting an appealing front while entering the values required by the developed machine learning model. The model was built on the IBM cloud platform.

CHAPTER 12
FUTURE WORKS

Since the dataset used for this model is an old vehicle dataset, the model's accuracy would suffer if the details of vehicles released recently were input. As a result, we propose that in the future, we use the most recent dataset set containing vehicle information to help train the model. We also intend to test other classification algorithms, such as SVM and Decision Tree, in place of Random Forest to see if any improvement in accuracy occurs. Finally, we propose that the machine learning model be scaled so that it can analyze the performance of a broader range of vehicles.

CHAPTER 13
APPENDIX

13.1 SOURCE CODE
13.1.1 car performance prediction.ipynb

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

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'vehicleperformanceanalyserdeploym-donotdelete-pr-zcujqjsilptifi'
object_key = 'car performance.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )


dataset = pd.read_csv(body)
dataset.head()
```

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

Normalisation

```
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()
x_train = sd.fit_transform(x_train)
x_test = sd.fit_transform(x_test)
y_train = sd.fit_transform(y_train)
y_test = sd.fit_transform(y_test)
```

```
x_train
```

Decision tree regressor

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(random_state=0,criterion="mae")
dt.fit(x_train,y_train)

import pickle
pickle.dump(dt,open('decision_model.pkl','wb'))

y_pred=dt.predict(x_test)
y_pred

y_test

ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred, 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()

from sklearn.metrics import r2_score,mean_squared_error
r2_score(y_test,y_pred)
0.8578094522360582
mean_squared_error(y_test,y_pred)
0.14219054776394183
np.sqrt(mean_squared_error(y_test,y_pred))
0.377081619498938
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
x11 = dataset.iloc[:,1:8].values
y11 = dataset.iloc[:,0].values
from sklearn.model_selection import train_test_split
x_train1, x_test1, y_train1, y_test1 = train_test_split(x11,y11,test_size=0.2,random_state=0)
rf= RandomForestRegressor(n_estimators=30,random_state=0)
```

```
rf.fit(x_train1,y_train1)
RandomForestRegressor(n_estimators=30, random_state=0)
y1_pred=rf.predict(x_test1)
y1_pred

from sklearn.metrics import r2_score
accuracy = r2_score(y_test1, y1_pred)
accuracy
0.8999792555413947
#save the model
import pickle
with open('car_performance_regression_pkl', 'wb') as files:
    pickle.dump(rf, files)

from sklearn.metrics import r2_score,mean_squared_error
r2_score(y_test,y_pred2)
-0.04347826086956519
mean_squared_error(y_test,y_pred2)
0.6
np.sqrt(mean_squared_error(y_test,y_pred2))
```

13.1.2 scoring end point.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 17 23:59:33 2022

@author: Max_l
"""

import requests

# NOTE: you must manually set API_KEY below using information retrieved from your
IBM Cloud account.
API_KEY = "isS3P7auiIh4rzYJVtIMforGUPRhkBUhxz1GPVFJ_MbV"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field":
[['cylinders','displacement','horsepower','weight','model year','origin']],
                                   "values": [[8,307,130,3504,70,1]]}]}

response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/947e6ad9-2c7b-4002-9bdf-
```

```
5e10aac95859/predictions?version=2022-11-17', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
```

13.1.3 app.py

```
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('RandomForestRegressor.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]) +"mpg. Carry extra
fuel"
    if(output>9 and output<=17.5):
        pred="Low performance with mileage " +str(prediction[0]) +"mpg. Don't go for long
distance"
    if(output>17.5 and output<=29):
        pred="Medium performance with mileage " +str(prediction[0]) +"mpg. Go for a ride
nearby."
    if(output>29 and output<=46):
        pred="High performance with mileage " +str(prediction[0]) +"mpg. Go for a healthy
ride"
    if(output>46):
        pred="That's a very high performance with mileage " +str(prediction[0])+"mpg. 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 trought 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)
```

13.1.4 index.html

```
<html>
<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
<link href="https://fonts.googleapis.com/css2?family=Girassol&display=swap"
rel="stylesheet">
<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

 <!-- <link rel="stylesheet" href="style.css"> -->

<div class="navbar">
      <section class="title">
      <h1>CAR PERFORMANCE PREDICTION</h1>
     </section>
</div>

<div class="wrapper fadeInDown">
  <div id="formContent">
    <!-- Tabs Titles -->
   <section class="date">
    <!-- Icon -->
    <div class="fadeIn first">
    </div>

    <form action="{{ url_for('y_predict')}}"method="post">
      <input type="text" name="Cylinders" placeholder="No.of cylinders (count)"
required="required" />
        <input type="text" name="Displacement" placeholder="Displacement (in miles)"
required="required" />
        <input type="text" name="Horsepower" placeholder="Horsepower (per sec)"
required="required" />
        <input type="text" name="Weight" placeholder="Weight (in pounds)"
required="required" />
        <input type="text" name="Model Year" placeholder="Model Year (YY)"
required="required" />
        <input type="text" name="Origin" placeholder="Origin" required="required" />
```

```
     <br>
    <input type="submit" class="fadeIn fourth" value="Predict">
  </form>
  </section>

  <div id="formFooter">
    <a class="underlineHover" href="#">
       <strong>{{ prediction_text }}</strong></a>
  </div>

 </div>
</div>
</html>
```

13.2 GitHub & Project Demo Link

Github : [Click here]()
Project Demo Link : [Click here]()