PROJECT REPORT

Date	18 November 2022
Team ID	PNT2022TMID25852
Project Name	Machine Learning based vehicle performance analyzer
Team Leader	Nareshkumar L
Team Members	Pradeep Kumar P M Nitin Yadav R Kamesh Kumar P

1. INTRODUCTION

The rapidly expanding discipline of data science includes machine learningas a key element. Algorithms are trained to generate classifications or predictions using statistical techniques, revealing important insights in data mining operations. The decisions made as a result of these insights influence key growthindicators in applications and enterprises, ideally. Data scientists will be more indemand as big data develops and grows because they will be needed to help identify the most important business issues and then the data to answer them. Machine learning is a subfield of artificial intelligence (AI) and computer sciencethat focuses on using data and algorithms to mimic human learning processes andprogressively increase accuracy.

1.1. PROJECT OVERVIEW

This project implies applied data science concepts and concepts of machine learning to the dataset that is collected as a part of the preparation phase to analyze and predict vehicle performance using a machine learning providing high accuracy and effficiency. In this project mulitple regression models are trained on the obtained dataset and checked for accuracy and tested for the suitability of the problem. Among different models that were trained decision tree regression have proven to be most effective with higher accuracy. Hence this model is saved for future predictions on vehicle performance given by the user which is obtained using a web application. This machine learning model is integrated with the web application with the help of a flask app. The entire application is also deployed on IBM cloud as a part of this project. On the other hand the project planning and management is implemented on Agile methodology where after each and every phase testing process is done. Here we perform performance and user acceptance testing after each sprint to ensure that the project stays on track and achieving its objectives on time.

1.2 PURPOSE

The main purpose of this project is to enable users to analyze their vehicle condition based on its performance. Predicting a car's performance level is a significant and intriguing challenge. Predicting a car's performance in order to change a certain behaviour of the vehicle is the major objective of the current study. This can drastically reduce the fuel consumption of the system

and boost efficiency. Analysis of the car's performance based on the horsepower, fuel type, engine type, and the number of cylinders. These are the variables that can be usedto forecast the condition of the vehicle. The process of gathering, investigating, interpreting, and documenting health data based on the aforementioned three elements is ongoing. For the prediction engine and engine management system, performance goals like mileage, dependability, flexibility, and cost can be integrated together and are crucial.

This strategy is a crucial first step in comprehending how a vehicle performs. Toincrease the performance efficiency of the vehicle, it is crucial to examine the elements utilising a variety of well-known machine learning methodologies, including as linear regression, decision trees, and random forests. Automobile engineering's "hot subjects" right now revolve around the power, longevity, and range of automotive traction batteries. We also take a performance in mileage into account here. We will create the models, utilising various techniques and neural networks, to resolve this issue. Then, we'll compare which algorithm accurately forecasts car performance (Mileage).

2. LITERATURE SURVEY

Here are some of the literature survey done to analyse different exiting models and to identify the right data and the machine learning model to approach the problem.

[1] Automotive Performance Tests Based on Machine Learning Algorithms

Authors:

- M. Geissler, IMST GmbH, Kamp-Lintfort, Germany
- J. Kunisch, IMST GmbH, Kamp-Lintfort, Germany
- C. Oikonomopoulos-Zachos, IMST GmbH, Kamp-Lintfort, Germany
- A. Friedrich, IMST GmbH, Kamp-Lintfort, Germany

Published on: 2022 16th European Conference on Antennas and Propagation

Abstract:

This paper suggests an innovative approach to define and perform tests of communication systems in cars. The test concept requires the placement of the vehicle under test on a planar turntable in an anechoic chamber. Software-defined multimode transceiver modules, referred to as radio heads, are placed in a quarter circle or half circle around the car at an adequate distance. This setup allows flexible, realistic, reproducible and dynamic over-the-air testing of the cars communication systems in the sense of a virtual drive test. One key topic - which is still open - is the definition of sufficiently realistic test scenarios related to real outdoor scenarios. The full description of those scenarios would require a prohibitively large number of parameters from the network and the channel to be considered, making it impractical to perform this derivation following a classical straight-forward approach. Therefore, this paper suggests the derivation of realistic test cases via a machine learning (ML) approach: instead of attempting to create a 1:1 mapping of real scenarios into the test chamber, we propose to use ML to identify and classify critical test cases via analysis of key performance indicators (KPI) of test data and from this to create representative synthetic test cases. This approach is currently under development and open for discussion here.

[2] Performance of Motor Vehicle based on Driving and Vehicle Data using Machine Learning

Authors:

- Nagaraje Gowda, Dublin, National College of Ireland**.**
- Punith Kumar, Dublin, National College of Ireland**.**

Published on: 2020, Masters thesis, Dublin, National College of Ireland.

Abstract:

With the increasing population demographics and the dependency of man on motor vehicles as the primary source of transportation, the number of motor vehicles being registered for commercial as well as non-commercial activities on a daily basis is massive and yet continues to increase at an alarming rate. This has a direct and an unambiguous effect on the amount of fossil fuels being utilized globally and its subsequent environmental effects, which is of great concern in the present situation. Several attempts from various research sectors are ongoing in order to overcome this global issue and promising results are expected. This project is one such attempt at identifying the performance of small passenger cars in terms of fuel efficiency and map them with factors affecting it using machine learning techniques. The commencing activity while carrying out any such research activity will be the identification of the problem and all its possible sources. In this case, two potential sources can be identified and they are; the vehicle characteristics and the driver/driving behaviour. The relevant data for this analysis was taken from the public source, Kaggle which is the data collected from the OBD of the car and models are built using techniques like Multiple Linear Regression, XGBoost, Support Vector Machine and Artificial Neural Network and their performance is compared to discover the first-rate technique in predicting the fuel efficiency and to propose the optimum driving behaviour in terms of throttle position to achieve better fuel efficiency. The results reveal that XGBoost model outperforms all other models developed in predicting the fuel efficiency for the different split ratios evaluated and comparing the throttle position

with the predicted fuel efficiency explains that to achieve better fuel efficiency the throttle position must be around 70 to 80 on a scale of 100, referred to as full throttle position. The knowledge discovered from the research could be used by car manufacturers to design cars in future to mitigate the fuel consumption.

[3] Transmission system performance analysis of traditional power vehicle

Authors:

- Feng Kang, Research Center of Advanced Powertrain Technology, State Key Laboratory of Advanced D&M for Vehicle Body, Hunan University, Changsha, China
- Liu Jingping, Research Center of Advanced Powertrain Technology, State Key Laboratory of Advanced D&M for Vehicle Body, Hunan University, Changsha, China
- Fu Jianqin, Research Center of Advanced Powertrain Technology, State Key Laboratory of Advanced D&M for Vehicle Body, Hunan University, Changsha, China
- Yang Hanqian, Research Center of Advanced Powertrain Technology, State Key Laboratory of Advanced D&M for Vehicle Body, Hunan University, Changsha, China

Published on: 2020, Masters thesis, Dublin, National College of Ireland.

Abstract:

Based on simulation software GT-drive, the author analyzed the transmission system performance of a passenger car with diesel engine and provided the appropriate research methods. Firstly, the numerical simulation model of a vehicle was built based on vehicle weight, frontal area, rolling, air-drag coefficient, etc. The different matching schemes were simulated and compared. The results show that, for a given engine, using different transmission systems, the matching efficiency is significantly different. In view of power and economy of the vehicle, it is important that selected suitable power transmission device. This method has provided a theoretical basis for studying traditional power vehicle, also giving some information to study the new type vehicle power train system.

[4] Steering performance simulation of three-axle vehicle with multi- axle dynamic steering

Authors:

- Shufeng Wang, College of Transportation and Vehicle Engineering, Shandong University of Technology, Zibo, China
- Junyou Zhang, College of Transportation and Vehicle Engineering, Shandong University of Technology, Zibo, China
- Huashi Li, College of Transportation and Vehicle Engineering, Shandong University of Technology, Zibo, China

Published on: 2008 IEEE Vehicle Power and Propulsion Conference

Abstract:

Because three-axle heavy-vehicle with front-wheel steering has big radius at low speed and bad stability at high speed, in order to improve heavy vehicle steering performance at different speed, the multi-axle dynamic steering technology is put forward. Selecting zero side-slip angle of mass center and proportional control strategy to control vehicle, Using MATLAB, the steering performance of the three-axle vehicle with different steering modes are simulated. The result shows that multi-axle dynamic steering can decrease the steering radius at low speed and improve vehicle stability at high speed.

2.1 EXISTING PROBLEM

The main problem in predicting machine learning is that different model suits for different situations but identifying which model suits overall with only some loss is crucial. So we need to train different regression models on the given data and identify which model suits our particular scenario and dataset so that it could be carried on for predicting purposes.

2.2 REFERENCES:

Byerly, A., Hendrix, B., Bagwe, R., Santos, E. and Ben-Miled, Z. (2019). A machine learning model for average fuel consumption in heavy vehicles, IEEE Transactions on Vehicular Technology, 68(7), 6343-6351, doi: 10.1109/TVT.2019.2916299 .

C¸ apraz, A. G., Ozel, P., S¸evkli, M. and Beyca, ¨ O. F. (2016). Fuel Consumption Models ¨Applied to Automobiles Using Real-time Data: A Comparison of Statistical Models, 83, pp. 774-781, doi: 10.1016/j.procs.2016.04.166.

Cortes, C. and Vapnik, V. (1995). Support-vector networks, Machine learning, 20(3), pp.273-297 . Fayyad, U. M., Haussler, D. and Stolorz, P. E. (1996). Kdd for science data analysis: Issues and examples., KDD pp. 50-56.

Freedman, D. A. (2009). Statistical models: theory and practice, cambridge university press. Fugiglando, U., Massaro, E., Santi, P., Milardo, S., Abida, K., Stahlmann, R., Netter, F. and Ratti, C. (2019). Driving behavior analysis through can bus data in an uncontrolled environment, IEEE Transactions on Intelligent Transportation Systems. 20(2), pp. 737- 748, doi: 10.1109/TITS.2018.2836308 .

Gilman, E., Keskinarkaus, A., Tamminen, S., Pirttikangas, S., R¨oning, J. and Riekki, J. (2015). Personalised assistance for fuel-efficient driving, Transportation Research Part C: Emerging Technologies, 58, pp. 681-705 doi: 10.1016/j.trc.2015.02.007 .

2.3 PROBLEM STATEMENT DEFNITION

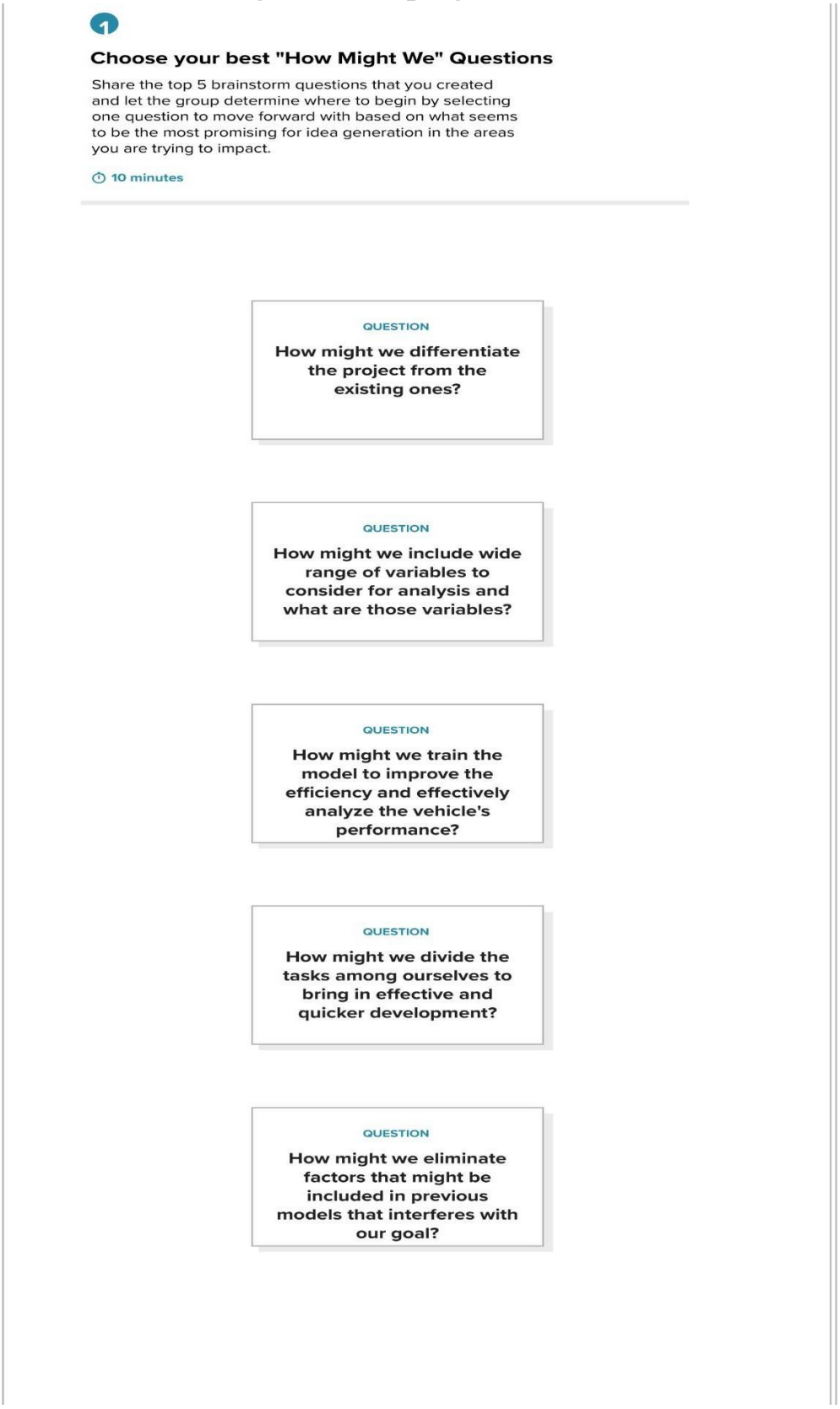
I am	**I am trying to**	**But**	**Because**	**Which makes me feel**
A daily driver	Evaluate performance	I don't know which aspect to focus more	A very large of factors impact performance	Confused
A regular person	To buy a new car	Can't focus on every single car	There are a variety of cars at different prices	Helpless

3. IDEATION AND PROPOSED SOLUTION:

3.1. EMPATHY MAP CANVAS

1.1IDEATION & BRAINSTORMING:

Step-1: Team Gathering, Collaboration and Select the Problem Statement
Step-2: Brainstorm, Idea Listing and Grouping



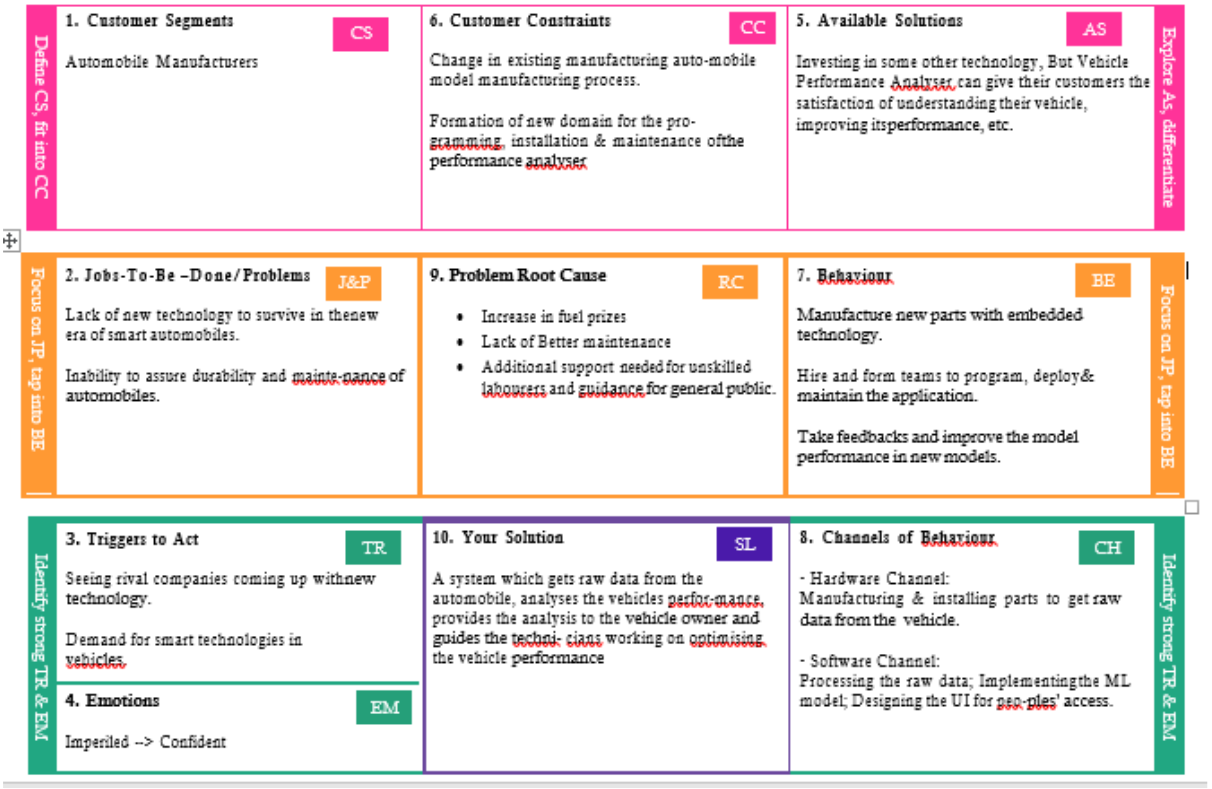
3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem tobe solved)	Predicting the performance level To efficiently improve the systems fuel consumption For efficient engine management system to improvethe mileage, dependability, flexibility.
2.	Idea / Solution description	Regression Model Algorithm will give more efficient solution to predict higher mpg. For which it should have higher R-squared value Also have smallest MSE value.
3.	Novelty / Uniqueness	By using decision tree, we can improve range, durability and longevity of automotive of batteries. It predicts performance Mileage with higher accuracy and efficiency.
4.	Social Impact / Customer Satisfaction	When the vehicle is in good condition with higher performance, it avoids the issues like emitting large amount of smoke that prevent air from getting polluted. Customer satisfaction – customer feels comfortableand smoothness while driving.
5.	Business Model (Revenue Model)	Revenue generation through selling your product asan application. Revenue generation through collaboration with car companies.
6.	Scalability of the Solution	The model will help the customers to choose wisely the vehicle as per their feasibility and learn more about their vehicle.

3.4 PROBLEM SOLUTION FIT:

Project Design Phase-I Solution Fit

Project Title: Machine Learning based Vehicle Performance Analyzer **Team ID: PNT2022TMID25852**



4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

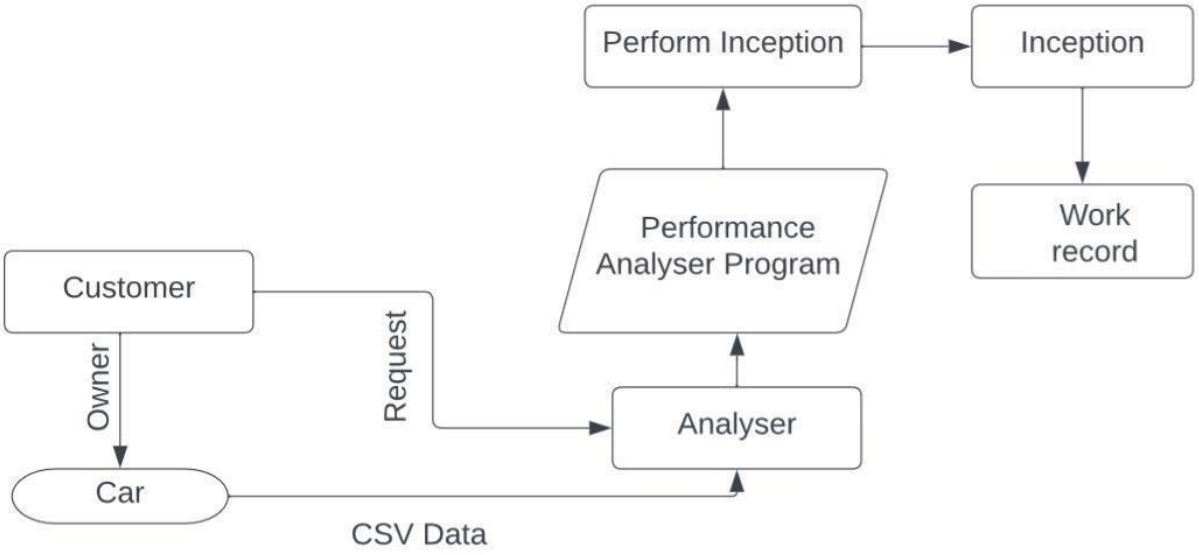
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Enters the inputs	Get Inputs through form
FR-2	User Essential	Predict the performance of the vehicle
FR-3	Data Prepossessing	Sample Dataset for training purpose
FR-4	User input Evaluation	Evaluating the given user values
FR-5	Prediction	Fuel consumption and efficiency of the vehicle

4.2 NON-FUNCTIONAL REQUIREMENTS:

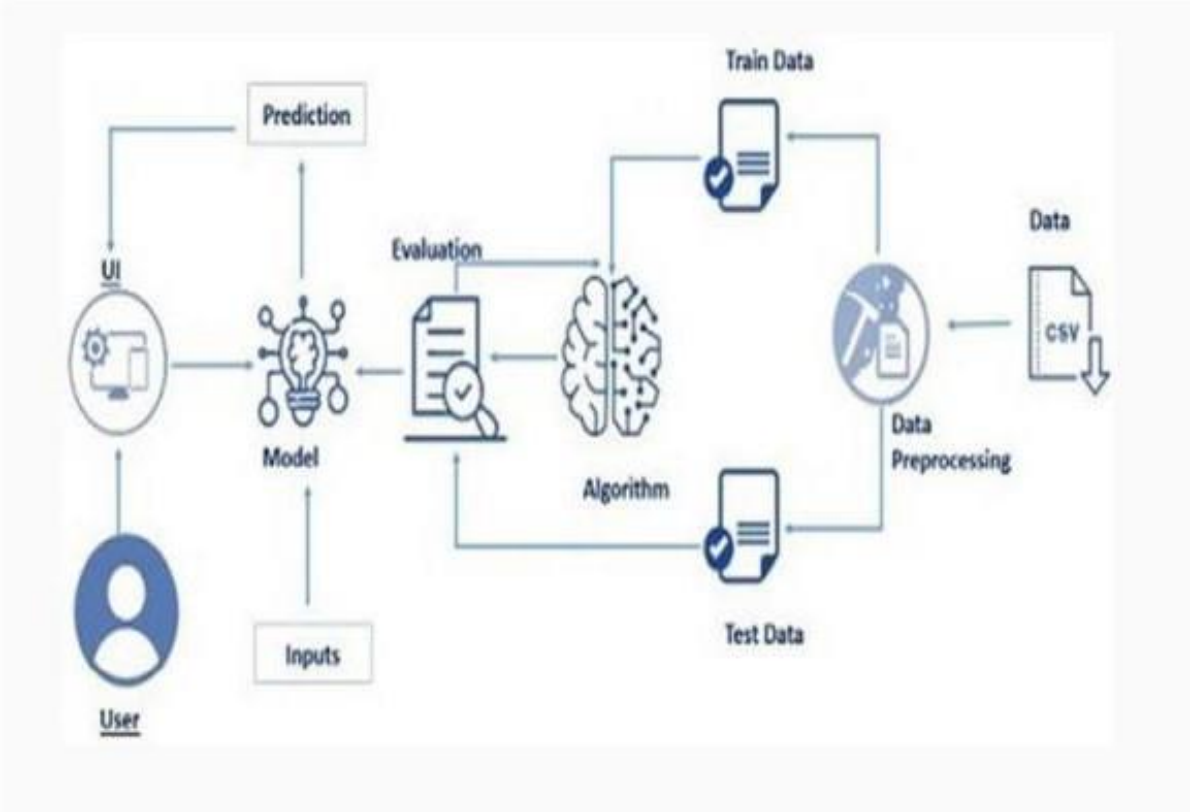
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The analyzer allows the user to improve performance based on the results provided. It is easy to use with just the data required.
NFR-2	**Security**	The security is improved by using vehicle alarm, wheel lock, vehicle lock and also GPS tracker.
NFR-3	**Reliability**	The reliability rating is good due to best performance, less frequency of problem occurrence and cost for repairing is low.
NFR-4	**Performance**	The vehicle is upgraded in their quality and infrastructure to provide better performance like good mileage, smooth travel.
NFR-5	**Availability**	The data required is collected by research persons and this data can be used to provide better results.
NFR-6	**Scalability**	Better scalability since our model analyses all information provides better refined solution. With less change to the vehicle, we could achieve maximum performance.

5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:



5.2 SOLUTION & TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint -1
Customer	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint -1
Customer	Registration	USN-3	As a user, I can register for the application through Online	I can register & access the dashboard with Facebook Login	Medium	Sprint -1
Customer	Login	USN-4	As a user, I can register for the application through Gmail	For vehicle performance prediction	High	Sprint -1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Dashboard	USN-5	As a user, I can log into the application by entering email & password	I will have knowledge about the vehicle performance	Medium	Sprint-2
Administrator	Login	USN-6	View vehicle performance	I can access my admin dashboard	High	Sprint-1
Administrator	Query Processing	USN-7	Try to rectify the query	Admin view the performance	Medium	Sprint-2

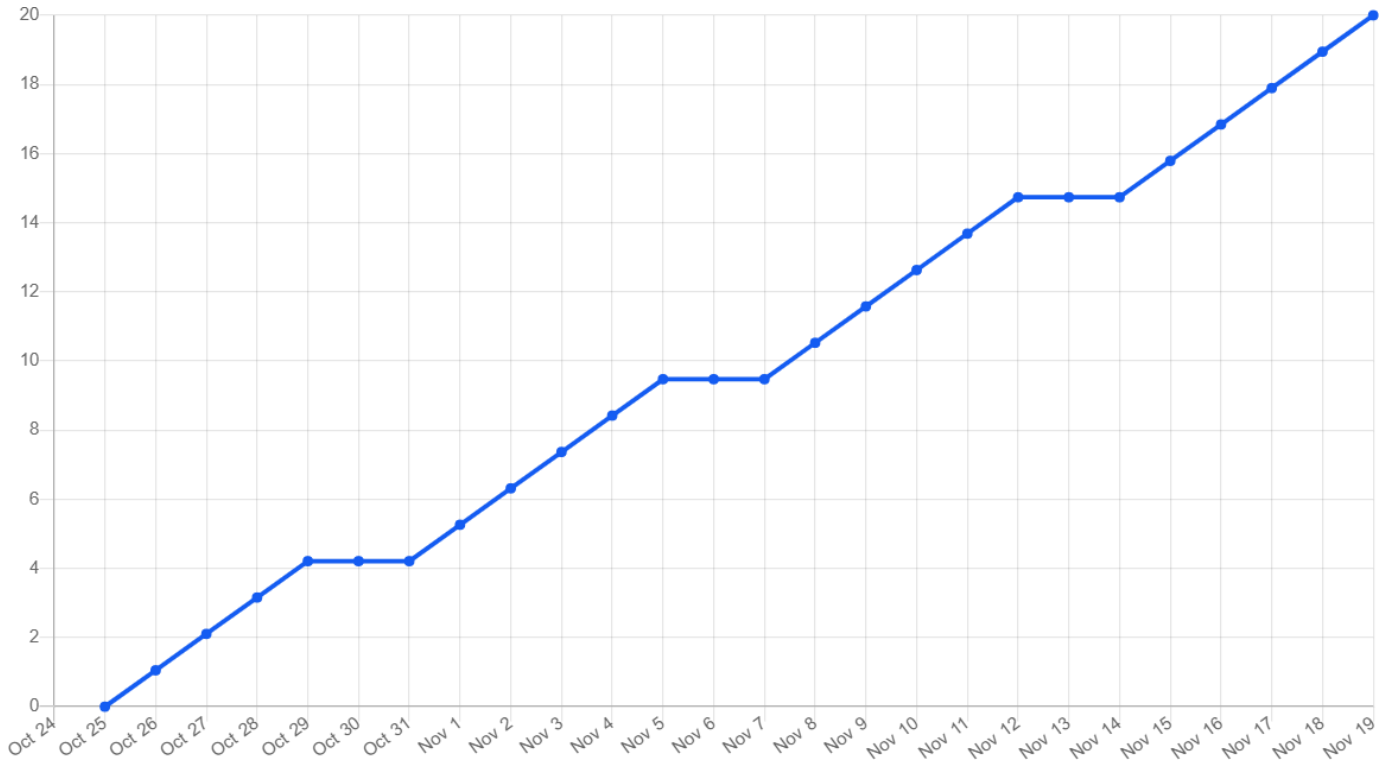
6.1. SPRINT PLANNING &ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Download the dataset	20	High	1
Sprint-2	Data Pre-processing	USN-2	Import libraries and read the dataset	4	Medium	1
Sprint-2		USN-3	Handle the missing value and label theencoding	4	Medium	2
Sprint-2		USN-4	Split the dataset into Dependent and independent variables	6	Medium	3
Sprint-2		USN-5	Split the dataset into train and test data	6	Medium	4
Sprint-3	Model Building	USN-6	Train the datasets to run smoothly and see an incremental improvement in the prediction rate for the available Machine Learning algorithms.	5	Low	1
Sprint-3		USN-7	Build The Model With The Decision TreeAlgorithm	6	Low	2
Sprint-3		USN-8	Predict The Values	5	Low	3
Sprint-3		USN-9	Model Evaluation	4	Low	4
Sprint-4	Application Building	USN-10	Building An Index. Html File	5	Low	1,2,3,4

6.1. 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	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

6.3 REPORTS FROM JIRA:



7. CODING & SOLUTIONING:

7.1 FEATURE 1:

The proposed system here is a machine learning based vehicle performance analyzer where users can enter different attributes associated with a vehicle and obtain its performance in this case mileage per gallon of fuel instantly. To choosea optimized machine learning algorithm we train most of the regression models to find the most suitable to approach this prediction. Among all the different models used it is inferred that decision tree regression seems to score high in performance metrics hence is chosen as the model to predict vehicle performance.Once the model is built and trained and when becomes ready for prediction, the model is dumped in a pickle file which can then be imported in application that requires it. Then create a flask application to use this pickled model to predict vehicle performance. On the other hand instead of pickling the model is deployedon IBM cloud and imported using the API key.

The below code is the model for Decision tree Algorithm.

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

The model is dumped into pickle and can be used as show belowimport

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

The flask app is created to act as an interface to predict the quality from the details the user is giving,

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('decision_model.pkl', 'rb'))



@app.route('/')
def home():
    return render_template('index.html')



@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    x_test = [[int(x) for x in request.form.values()]]
    print(x_test)
    #sc = load('scalar.save')
    prediction = model.predict(x_test)
    print(prediction)
    output=prediction[0]
    if(output<=9):
        pred="Worst performance with mileage " + str(prediction[0]) +". Carry extra fuel"
    if(output>9 and output<=17.5):
        pred="Low performance with mileage " +str(prediction[0]) +". Don't go to long
distance"
```

```
    if(output>17.5 and output<=29):
        pred="Medium performance with mileage " +str(prediction[0]) +". Go for a ride nearby."
    if(output>29 and output<=46):
        pred="High performance with mileage " +str(prediction[0]) +". Go for a healthy ride"
    if(output>46):
        pred="Very high performance with mileage " +str(prediction[0])+". You can plan for a Tour"



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

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls through request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
```

7.2 FEATURE 2:

On the other hand instead of dumping into a pickle file, the model can be deployed in the IBM Cloud and can be used using the API key.

```
import requests
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "d45n5LrUHP4VF_ApgbRYyERhNatqy16RtVnjk55hXMUp"
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": [{"fields": [['f0','f1','f2','f3','f4','f5']], "values":
[[8,160,380,3504,82,1]]}]}

response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/9bd1d84b-810a-4086-a522-
1e4c8d08a1da/predictions?version=2022-11-18', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
pred=response_scoring.json()
output=pred['predictions'][0]['values'][0][0]
print(output)
```

8. TESTING:

- Verify that the user could able to use that web page.
- Verify that the user could able to enter the value.
- Verify that the values entered by the user are computed.
- Verify that the user could able to see the predicted value.

8.2. USER ACCEPTANCE TESTING:

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

Resolution	**Severity 1**	**Severity 2**	**Severity 3**	**Severity Subtotal**	
By Design	7	4	2	3	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	15	31
Not Reproduced 0		0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	3	2	1	6
Totals	20	12	13	21	66

2. 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	6	0	0	6
Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	2	0	0	2
Exception Reporting	7	0	0	7
Final Report Output	3	0	0	3
Redirecting	1	0	0	1

9. RESULTS

9.1. PERFORMANCE METRICS:

Here to predict the performance of the above model two main measures are used. Model Accuracy and the r-square value. Then the mean squared error for the value is also checked. R-squared is a statistical measure of how close the data areto the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. The R- squared value for the model is 0.912578781275149 and the value of mean squareerror is 6.042499999999999.

10. ADVANTAGES AND DISADVANTAGES:

Advantages:

- The model enables an user to immediately analyze a vehicle's performanceand provide results instantly.
- The model uses decision tree regression which is proved to be moresuitable for such cases.
- The model takes into account various error factors and acts upon them toproduce almost accurate results.
- It automates the tedious and repetitive tasks.

Disadvantages:

- When dimension of the data is high the model tends to take little more time.
- This model is only suitable for measuring performance in terms of miles per gallons, and might not be suitable for other performance measure suchas comfort etc.

11. CONCLUSION:

Vehicle performance prediction by using this model becomes easy and simple. It enables users of all category to predict their vehicle's performance without needing a deeper knowledge of know how about the vehicle. By employing thiscustomers can also decide to sell or buy vehicles and it makes this transaction easier and clearer. The above model that is decision tree regression used is verymuch suitable to this scenarios and has an accuracy of about 98.06333123. It is on an overall scale doing good keeping prediction closer to accurate values.

12. FUTURE SCOPE:

The scope for this project is quite high due to high scalable nature. As almost everyone in the world owns a vehicle and everyone wants to know how their vehicle performing. This is a global scale and task which can be fulfilled using this model. The scalable and reliable nature based on its accuracy provides the clearance for the model to be employed everywhere for vehicle performance prediction.

13. APPENDIX:

SOURCE CODE:

Vehicle_performance_analysis.ipynb

#!/usr/bin/env python#

coding: utf-8

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

Finding missing data

dataset.isnull().any()

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

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

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

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

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

dataset.info() #Pandas dataframe.info() function is used to get a quick overviewof the dataset.

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

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

Data Visualizations

Heatmap : which represents correlation between attributes

sns.heatmap(dataset.corr(),annot=True,linecolor ='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 relationacross the entire dataframe.

plt.show()

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

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

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

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

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

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

sns.regplot(x="model year", y="mpg", data=dataset)

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

sns.set(style="whitegrid") sns.boxplot(x=dataset["mpg"])

# Finding quartiles for mgp

# # 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 are95% confident that the correlation between

# the variables is significant.#

# By convention, when the#

<ul>

#           <li>p-value is $<$ 0.001: we say there is strong evidence that the correlation is significant.</li>

#           <li>the p-value is $<$ 0.05: there is moderate evidence that the correlation is significant.</li>

#           <li>the p-value is $<$ 0.1: there is weak evidence that the correlation is significant.</li>

#           <li>the p-value is $>$ 0.1: there is no evidence that the correlation is significant.</li>

# </ul>
```

```
from scipy import stats


# <h3>Cylinders vs mpg</h3>#

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $<$ 0.001, the correlation between cylinders and mpgis statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.

# <h3>Displacement vs mpg</h3>#

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $<$ 0.1, the correlation between displacement andmpg is statistically significant, and the linear negative relationship is quite strong (~-0.809, close to -1)</p>

# <h3>Horsepower vs mpg</h3>

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $<$ 0.001, the correlation between horsepower andmpg is statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative and moderately strong.

# <h3>Weght vs mpg</h3>

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.001, the correlation between weight and mpg is statistically significant, and the linear negative relationship is quite strong (~- 0.831, close to -1)</p>

# <h3>Acceleration vs mpg</h3>#

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $>$ 0.1, the correlation between acceleration and mpg is statistically significant, but the linear relationship is weak (~0.420).</p>

# <h3>Model year vs mpg</h3>
```

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-valueof P =", p_value)

<h5>Conclusion:</h5>

Since the p-value is $<$ 0.001, the correlation between model year andmpg is statistically significant, but the linear relationship is only moderate (~0.579).

<h3>Origin vs mpg</h3>#

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-valueof P =", p_value)

<h5>Conclusion:</h5>

<>Since the p-value is $<$ 0.001, the correlation between origin and mpg is statistically significant, but the linear relationship is only moderate (~0.563).

Ordinary Least Squares Statistics

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

test.summary()

Inference as in the above summary the p value of the accelaration is maximum(i.e 0.972) so we can remove the acc variable from the dataset

Seperating into Dependent and Independent variables#

Independent variables

```
x=dataset[['cylinders','displacement','horsepower','weight','model
year','origin']].values

x

# <b>Dependent variables</b>

y=dataset.iloc[:,0:1].values

y

# # 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

# # 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

import os

os.environ['PATH'] =
os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\graphviz "

from sklearn.externals.six import StringIOfrom

IPython.display import Image

from sklearn.tree import export_graphviz
```

```
import pydotplus

dot_data = StringIO() export_graphviz(dt,

out_file=dot_data,

filled=True, rounded=True,

special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png()

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()

# 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 roomfor improvement.

# <b>R-squared</b>

# <p>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 ofmultiple determination for multiple regression.</p>

#
```

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).

from sklearn.metrics import r2_score,mean_squared_error

r2_score(y_test,y_pred) mean_squared_error(y_test,y_pred)

np.sqrt(mean_squared_error(y_test,y_pred))

random forest regressor

from sklearn.ensemble import RandomForestRegressor

rf= RandomForestRegressor(n_estimators=10,random_state=0,criterion='mae')

rf.fit(x_train,y_train)

y_pred2=rf.predict(x_test)y_pred2

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 roomfor improvement.

```
from sklearn.metrics import r2_score,mean_squared_error

r2_score(y_test,y_pred2)

mean_squared_error(y_test,y_pred2)

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

linear regression

from sklearn.linear_model import LinearRegression

mr=LinearRegression()

mr.fit(x_train,y_train)

y_pred3=mr.predict(x_test)

y_pred3

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

sns.distplot(y_pred3, 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 not as close to the actual values, since thetwo
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_pred3) mean_squared_error(y_test,y_pred3)

np.sqrt(mean_squared_error(y_test,y_pred3))

# <b>Conclusion:</b>
```

When comparing models, the model with the higher R-squared value is a better fit for the data.

When comparing models, the model with the smallest MSE value is a better fit for the data.

#

Comparing these three models, we conclude that the DecisionTree model isthe best model to be able to predict mpg from our dataset.

#

Vehicle_performance_Analysis_IBM_Deployment.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

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

dataset

Finding missing data

dataset.isnull().any()

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

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

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

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


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

dataset.info() #Pandas dataframe.info() function is used to get a quick overviewof
the dataset.

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

# 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


# # Data Visualizations

# Heatmap : which represents correlation between attributes

sns.heatmap(dataset.corr(),annot=True,linecolor ='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 relationacross the entire dataframe.

plt.show()

# Regression plots(regplot()) creates a regression line between 2 parameters andhelps to visualize their linear relationships.

sns.regplot(x="cylinders", y="mpg",

data=dataset) sns.regplot(x="displacement",

y="mpg", data=dataset)

sns.regplot(x="horsepower", y="mpg",

data=dataset) sns.regplot(x="weight",

y="mpg", data=dataset)

sns.regplot(x="acceleration", y="mpg",

data=dataset) sns.regplot(x="model year",

y="mpg", data=dataset)

sns.regplot(x="origin", y="mpg",

data=dataset) sns.set(style="whitegrid")

sns.boxplot(x=dataset["mpg"])

# Finding quartiles for mgp

# # The P-value is the probability value that the correlation between these twovariables is statistically significant.

# Normally, we choose a significance level of 0.05, which means that we are95% confident that the correlation between

# the variables is
```

```
significant.#

# By convention,

when the# <ul>

#           <li>p-value is $<$ 0.001: we say there is strong
evidence that thecorrelation is significant.</li>

#           <li>the p-value is $<$ 0.05: there is moderate evidence that
the correlationis significant.</li>

#           <li>the p-value is $<$ 0.1: there is weak evidence that the
correlation issignificant.</li>

#           <li>the p-value is $>$ 0.1: there is no evidence that the
correlation issignificant.</li>

# </ul>


from scipy import stats

# <h3>Cylinders vs

mpg</h3>#

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>
```

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

<h3>Displacement vs

mpg</h3>#

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-valueof P =", p_value)

<h5>Conclusion:</h5>

Since the p-value is $<$ 0.1, the correlation between displacement andmpg is statistically significant, and the linear negative relationship is quite strong (~-0.809, close to -1)

<h3>Horsepower vs mpg</h3>

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-valueof P =", p_value)

<h5>Conclusion:</h5>

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

```
# <h3>Weght vs mpg</h3>

# 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-valueof P =", p_value)


# <h5>Conclusion:</h5>

# <p>Since the p-value is $<$ 0.001, the correlation between weight
and mpg isstatistically significant, and the linear negative relationship is
quite strong (~- 0.831, close to -1)</p>

# <h3>Acceleration vs mpg</h3>#


# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $>$ 0.1, the correlation between acceleration
and mpg is statistically significant, but the linear relationship is weak
(~0.420).</p>

# <h3>Model year vs mpg</h3>
```

```
# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>

# <p>Since the p-value is $<$ 0.001, the correlation between model
year andmpg is statistically significant, but the linear relationship is
only moderate (~0.579).</p>

# <h3>Origin vs mpg</h3>
#

# 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-valueof P =", p_value)

# <h5>Conclusion:</h5>


# <>Since the p-value is $<$ 0.001, the correlation between origin and
mpg isstatistically significant, but the linear relationship is only
moderate (~0.563).</p>

# <b>Ordinary Least Squares</b>  Statistics

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

```
test.summary()

# Inference as in the above summary the p value of the accelaration
is maximum(i.e 0.972) so we can remove the acc variable from the
dataset

# # Seperating into Dependent and Independent

variables# <b>Independent variables</b>

x=dataset[['cylinders','displacement','horsepower','weight','model
year','origin']].values

x

# <b>Dependent variables</b>y=dataset.iloc[:,0:1].

Values y


# # 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

# # decision tree regressor

from sklearn.tree import DecisionTreeRegressor

dt=DecisionTreeRegressor(random_state=0,criterion="mae")

dt.fit(x_train,y_train)

y_pred=dt.predict(x_te

st)y_pred
```

```
y_test import os

os.environ['PATH'] =
os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\
graphviz "

from sklearn.externals.six import

StringIOfrom IPython.display

import Image

from sklearn.tree import

export_graphvizimport

pydotplus

dot_data = StringIO()

export_graphviz(dt,

out_file=dot_data,

filled=True, rounded=True,

special_characters=True)

graph =

pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png()

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()

# 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 roomfor improvement.

# <b>R-squared</b>

# <p>R-squared is a statistical measure of how close the data are to
the fittedregression line.

# It is also known as the coefficient of determination, or the
coefficient ofmultiple determination for multiple regression.</p>

#

# R-squared = Explained variation / Total

variation# <b>Mean Squared Error

(MSE)</b>

# <p>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).</p>


from sklearn.metrics import

r2_score,mean_squared_error

r2_score(y_test,y_pred)

mean_squared_error(y_test,y_pred)

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

```
# # random forest regressor
from sklearn.ensemble import RandomForestRegressor
rf= RandomForestRegressor(n_estimators=10,random_state=0,criterion='mae')
rf.fit(x_train,y_train)
y_pred2=rf.predict(x_t
est)y_pred2
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('Proportio 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 roomfor improvement.
from sklearn.metrics import
r2_score,mean_squared_error
r2_score(y_test,y_pred2)
mean_squared_error(y_test,y_pred2)
np.sqrt(mean_squared_error(y_test,y_pred2))# # linear regression
```

```
from sklearn.linear_model import
LinearRegressionmr=LinearRegression()
mr.fit(x_train,y_tra
in)
y_pred3=mr.predic
t(x_test)y_pred3
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred3, 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 not as close to the actual values,
since thetwo 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_pred3)
mean_squared_error(y_test,y_pred3)
np.sqrt(mean_squared_error(y_test,y_pred3))
```

```
from ibm_watson_machine_learning import
APIClientwml_credentials = {
'apikey' : "YIJAXb1Vp23FVn6FxaWNfEECIbjRwptpHaaL7jNGzuTE",

"url" : "https://eu-gb.ml.cloud.ibm.com"
}
wml_client=APIClient(wml_credentials)
wml_client.spaces.list()
space_id="42b68706-c255-41ca-87bf-bbafc459a92c"
wml_client.set.default_space(space_id)
wml_client.software_specifications.list()
model_name="analysis_model"
deployment_name="analysis_deploy_model"
model_deploy=dt
software_spec_uid=wml_client.software_specifications.get_uid_by_na
me("runt ime-22.1-py3.9")
model_props={
wml_client.repository.ModelMetaNames.NAME:model_na
me, wml_client.repository.ModelMetaNames.TYPE:"scikit-
learn_1.0",
wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software
_spec_uid
}
model_details=wml_client.repository.store
_model(model=model_deploy,
meta_props=model_props,
training_data=x_train,
training_target=y_train)model_details
```

Index.html

```
<link
                href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/boots
trap.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') }}">
<div class="navbar">
<section class="title">
<h1><p style="font-family: 'cursive', cursive ;">PREDICT YOUR CAR'S
PERFORMANCE</p></h1>
</section>
</div>

<div class="wrapper fadeInDown">
<div id="formContent">
<!-- Tabs Titles -->
<section class="date">
<!-- Icon -->
<div class="fadeIn first">
<script src="https://unpkg.com/@lottiefiles/lottie-player@latest/dist/lottie-
```

```
player.js"></script>
</div>
<div class="Contanier">
<div class="card"></div>

</div>
<div class="fadeInDown">
<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
(inmiles)" required="required" />
<input type="text" name="Horsepower" placeholder="Horsepower
(persec)" 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>
</div>
```

app.py

```
import numpy as np
from flask import Flask, request, jsonify,
render_templateimport pickle
#from joblib
import loadapp
= Flask(__________name__)
model = pickle.load(open('decision_model.pkl', 'rb'))

@app.
route('
/')def
home(
):
return render_template('index.html')


@app.route('/y_predict',methods=['P
OST'])def y_predict():
'''
For rendering results on
HTML GUI'''
```

```
x_test = [[int(x) for x in
request.form.values()]]print(x_test)
#sc =  load('scalar.save')
prediction =
model.predict(x_test)
print(prediction)
output=prediction[0]
if(output<=9):
pred="Worst performance with mileage " + str(prediction[0]) +". Carry
extrafuel"
if(output>9 and output<=17.5):
pred="Low performance with mileage " +str(prediction[0]) +". Don't go
tolong distance"
if(output>17.5 and output<=29):
pred="Medium performance with mileage " +str(prediction[0]) +". Go
for aride nearby."
if(output>29 and output<=46):
pred="High performance with mileage " +str(prediction[0]) +". Go for
ahealthy ride"
if(output>46):
pred="Very high performance with mileage " +str(prediction[0])+". You
can plan for a Tour"


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

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

For direct API calls 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)

IBM_app.py

import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import requests

NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "YIJAXb1Vp23FVn6FxaWNfEECIbjRwptpHaaL7jNGzuTE"

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}
#from joblib import load
app = Flask(_____name__)

@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')
payload_scoring = {"input_data": [{"fields":
[['f0','f1','f2','f3','f4','f5']],"values": x_test }]}

response_scoring =
requests.post('https://eu-
gb.ml.cloud.ibm.com/ml/v4/deployments/f4aecc62-cd58-47a3-af62-
6a940301a611/predictions?version=2022-11-15', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response") print(response_scoring.json())
pred=response_scoring.json() output=pred['predictions'][0]['values'][0][0]
print(output)

if(output<=9):
ped="Worst performance with mileage " + str(output) +". Carry extra fuel"
if(output>9 and output<=17.5):
ped="Low performance with mileage " +str(output) +". Don't go to long
distance"
```

```
if(output>17.5 and output<=29):
ped="Medium performance with mileage " +str(output) +". Go for a ride
nearby."
if(output>29 and output<=46):
ped="High performance with mileage " +str(output) +". Go for a healthy
ride"
if(output>46):
ped="Very high performance with mileage " +str(output)+". You can planfor a
Tour"
return render_template('index.html', prediction_text='{}'.format(ped))

if_____name_== "__main_":app.run(debug=True)
```