PROJECT REPORT

MACHINE LEARNING BASED VEHICLE PERFORMANCE ANALYSER

TEAM ID : PNT2022TMID54018

TEAM SIZE : 4

TEAM LEADER : VINOTHINI M K

TEAM MEMBER : PETCHIAMMAL A

TEAM MEMBER : SREEDEVI S

TEAM MEMBER : VINISHA P

INTRODUCTION :

The rapidly expanding discipline of data science includes machine learning as 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 growth indicators in applications and enterprises, ideally. Data scientists will be more in demand 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 science that focuses on using data and algorithms to mimic human learning processes and progressively increase accurancy.

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 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.

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 used to 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. To increase 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).

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] An approach of modeling on dynamic performance evaluation for off- road vehicle

Authors:

Junshu Han

Institute of Medical Equipment, Academy of Military Medical Sciences, Tianjin,China

Zhenhai Gao

Institute of Medical Equipment, Academy of Military Medical Sciences, Tianjin,China

Shulin Tan

Institute of Medical Equipment, Academy of Military Medical Sciences, Tianjin,China

Xiangdong Cui

Institute of Medical Equipment, Academy of Military Medical Sciences. Tianjin,China

Published in: 2010 8th World Congress on Intelligent Control and Automation

Abstract:

Automobile dynamic is one of the most important performance indexes, the key is whether the simulation accords with the real status, whether the vehicle performance under real driving conditions can be reflected more validly, and it will be used to estimate automobile dynamic accurately, or to provide theory reference for vehicle design. On the point of view of using vehicle, considering the effects of external factors as air velocity, tyre slip, adhesion coefficient on vehicle dynamic performance, a novel method on dynamic performance evaluation is established, and the effectiveness of the dynamic performance model is vertified.

[2] 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 in: 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 centre 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.

[3] Simulation study on synthetical performance of electric vehicles

Authors:

Sun Fengchun

Electric Vehicle Research and Development Center, Beijing Institute of Technology, Beijing, China

Sun Liqiug

Zhu Jiaguang

Electric Vehicle Research and Development Center, Beijing Institute of Technology, Beijing, China

Published in: Proceedings of the IEEE International Vehicle Electronics Conference (IVEC'99) (Cat. No.99EX257)

Abstract:

This paper presents the software development on the performance simulation of electric vehicles. Software verification is carried out via the comparison of simulation results with on-road test. Applications of the software in prototype design are also presented in terms of theoretical inference, modeling, software development and simulation of synthetical performance for EVS such as dynamic

performance, economy performance as well as analysis of parameters' influences on EV performance. The commonly used European drive cycle is adopted for simulation in the paper. Simulation with the software proves an efficient and money-saving means for prototyping of EV or HEV systems with control units.

[4] Simulation and Analysis of Performance of a Pure Electric Vehicle with a Super-capacitor

Authors:

N Jinrui

School of mechanical and vehicular engineering, Beijing Institute of Technology

[5] Steering feel study on the performance of EPS

Authors:

Xin. Zhang
School of Mechanical, Electric and Control Engineering, Beijing Jiaotong University, Beijing, China

Zhang Xin
School of Mechanical, Electric and Control Engineering, Beijing Jiaotong University, Beijing, China

Shi Guobiao
Electric Vehicle Center of Analysis and Technology, Beijing Institute of Technology, Beijing, China

Published in: 2008 IEEE Vehicle Power and Propulsion Conference

Abstract:

The steering feel study is very important in the development of electric power steering system (EPS). This paper describes a method about how to evaluate and get the suitable steering feel when driving a vehicle equipped with EPS. The EPS steering feel subjective tests were performed to obtain objective quality parameters that correlate with subjective evaluation. Afterthis, the paper briefly describes the statistical technique used to identify which parameters best correlate with vehicle steering qualities. As there was no correlation between a single partial rating and a single objective indicator, the principal component analysis (PCA) method was chosen and obtained objective indices. The objective evaluation parameters have been validated by drivers psila subjective evaluation. In the third part, the analytical method was applied to vehicle dynamic analysis to analyze vehicle steering feel

characteristics, we established a closed-loop steering feel simulation model to analyze steering torque characteristics, vehicle dynamic response and assess steering feel performance for different settings of a EPS system. The design of EPS was optimized and achieved more suitable driving feel by using the dynamic analysis model without plenty of real vehicle tests. This method make it possible to easily and accurately benchmark steering dynamic characteristics, set design targets, and is helpful to achieve good steering feel.

[6] Study on the performance and control of SR machine for vehicle regenerative braking

Authors:

Xiaoling Yuan

College of Electrical Engineering, Hohai University, HHU, Nanjing, Jiangsu,China

Yimin Gao

Department of Electrical Engineering, Texas A and M University, College Station, TX, USA

M. Ehsani

Department of Electrical Engineering, Texas A and M University, College Station, TX, USA8

Published in: 2008 IEEE Vehicle Power and Propulsion Conference

Abstract:

A regenerative braking system with simple structure, high efficiency, good performance and easy control is crucial for electric vehicle (EV), hybrid electric vehicle (HEV) and fuel cell vehicle (FCV). SR machine is one of the promising candidates. In this paper, the current and torque performance of a SR machine for application to vehicle regenerative braking has been studied. The relationship between the torque, speed, turn-on and turn-off angles has been established. The data obtained through simulation is very useful for vehicle control design.

[7] A method to analyze driver influence on the energy consumption and power needs of electric vehicles

Authors:

Rayad Kubaisi

Chair of Vehicle Technology, Karlsruhe, Germany Institute of Vehicle System Technology,Frank Gauterin

Chair of Vehicle Technology, Karlsruhe, Germany Institute of Vehicle System Technology,Martin Giessler

Chair of Vehicle Technology, Karlsruhe, German

Published in: 2009 IEEE Intelligent Symposium Institute of Vehicle System Technology.

Abstract:

The energy consumption and power needs of electric vehicles are evaluated on roller test benches according to test procedures defined by legal standards and by vehicle manufacturers. These test procedures are mainly defined by driving cycles and include tolerances to compensate for the human error during these tests. These tolerances may seem to make the tests easier but they can have a big effect on the appropriate dimensioning of the components, and also on the performance of the vehicle. Within this paper, a method is presented, which enables the quantification of these effects depending on the type of the test procedure, and the way the driving cycle is driven. The developed method has been tested in a simulation environment and several standard test procedures were analyzed.
[8] 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 in: 2011 International Conference on Electric Information and Control Engineering

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.

[9] Real-time performance of control allocation for actuator coordination in heavy vehicles

Authors:

Kristoffer Tagesson

Department Chassis Strategies & Vehicle Analysis, VOLVO 3PDepartment 26661, AB4S GOTEBORG, Sweden
Peter Sundstrom
MaDELON AB, Lund, Sweden

Leo Laine
Department Chassis Strategies & Vehicle Analysis, VOLVO 3PDepartment 26661, AB4S GOTEBORG, Sweden

Published in: 2009 IEEE Intelligent Vehicles Symposium

Abstract:

This paper shows how real-time optimisation for actuator coordination, known as control allocation, can be a viable choice for heavy vehicle motion control systems. For this purpose, a basic stability control system implementing the method is presented. The real-time performance of two different control allocation solvers is evaluated and the use of dynamic weighting is analysed.

Results show that sufficient vehicle stability can be achieved when using control allocation for actuator coordination in heavy vehicle stability control. Furthermore, real-time simulations indicate that the optimisation can beperformed with the computational capacity of today's standard electronic control units

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:

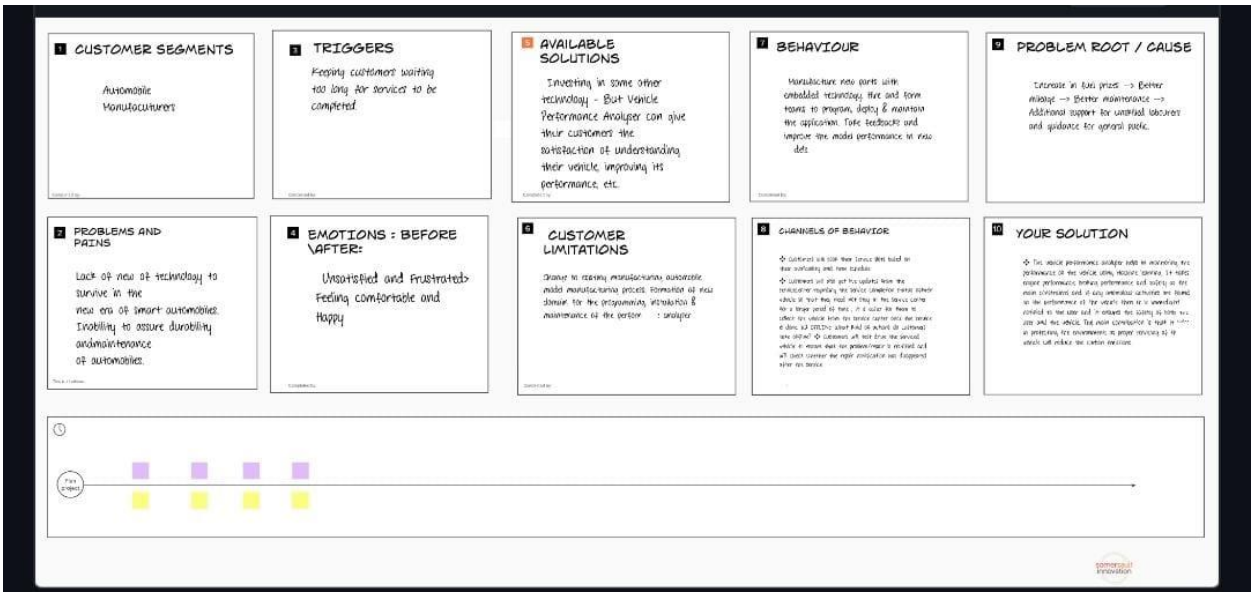
Byerly, A., Hendrix, B., Bagwe, R., Santos, E. and Ben-Miled, Z. (2019). A machine learning model for average fuel consumption in heavy vehicles, IEEE Transactions on Vehicular Technology, 68(7), 6343-6351, doi: 10.1109/TVT.2019.2916299 .

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.

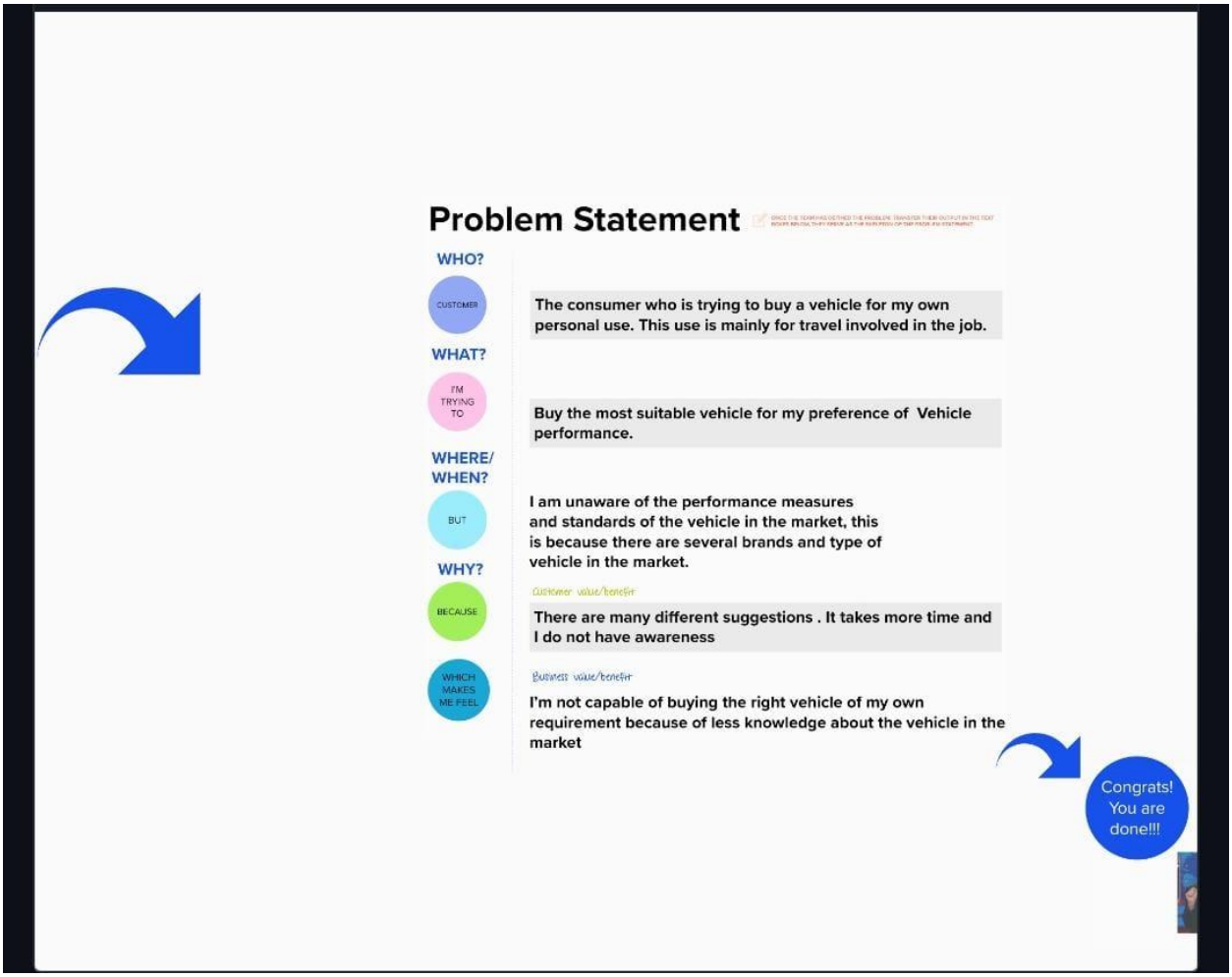
C. and Vapnik, V. (1995). Support-vector networks, Machine learning, 20(3), pp.273-297 . Fayyad, U. M., Haussler, D. and Stolorz, P. E. (1996). Kdd for science data analysis: Issues and examples., KDD pp. 50-56.Freedman, D. A. (2009). Statistical models: theory and practice, cambridge university press.

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 .

E., Keskinarkaus, A., Tamminen, S., Pirttikangas, S., R¨oning, J. and , 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.

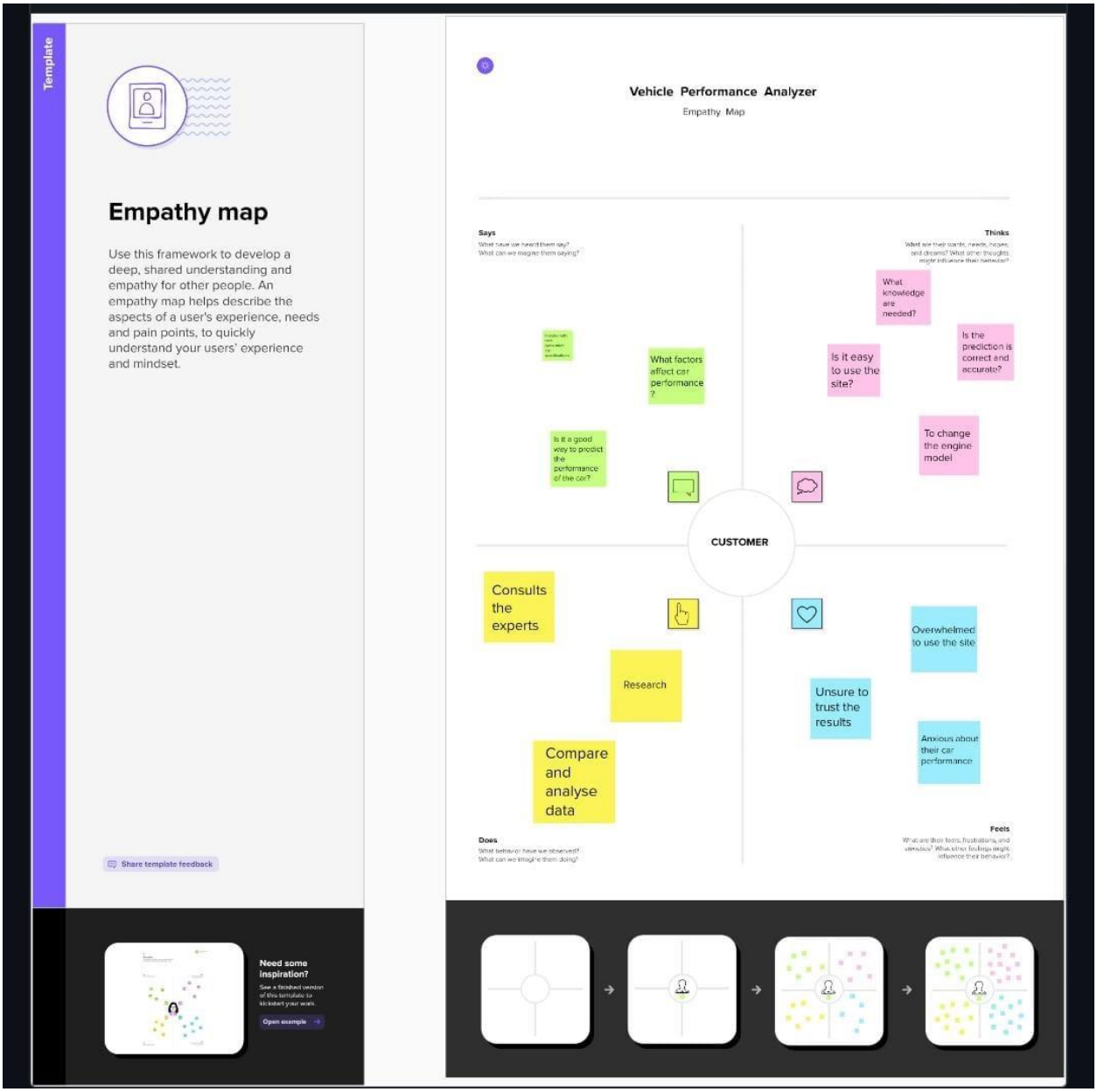


2.3 PROBLEM STATEMENT DEFNITION



3. IDEATION AND PROPOSED SOLUTION:

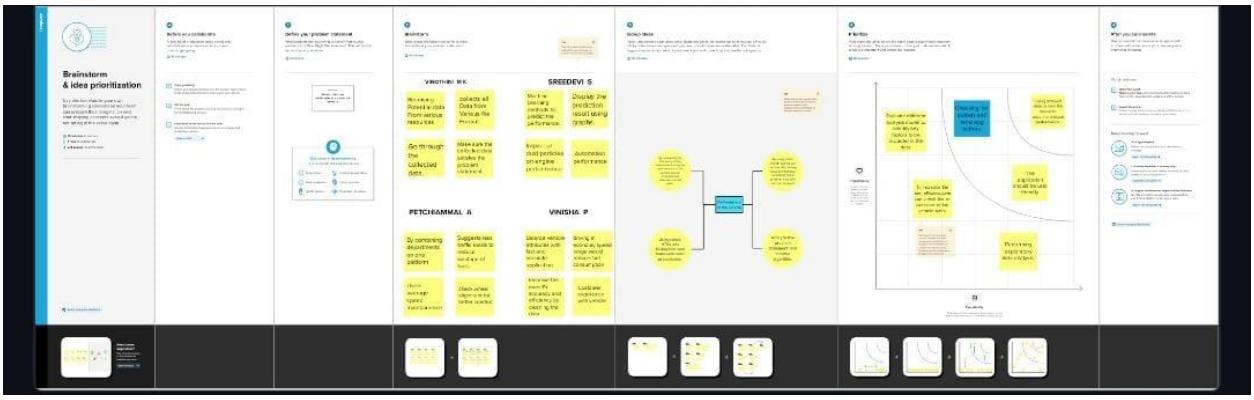
3.1 EMPATHY MAP CANVAS



IDEATION & BRAINSTORMING:

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping



3.3 PROPOSED SOLUTION

Project Design Phase-I

Proposed Solution Template

Date	21 October 2022
Team ID	PNT2022TMID54018
Project Name	Project - Machine Learning based Vehicle Performance Analyzer
Maximum Marks	2 Marks

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

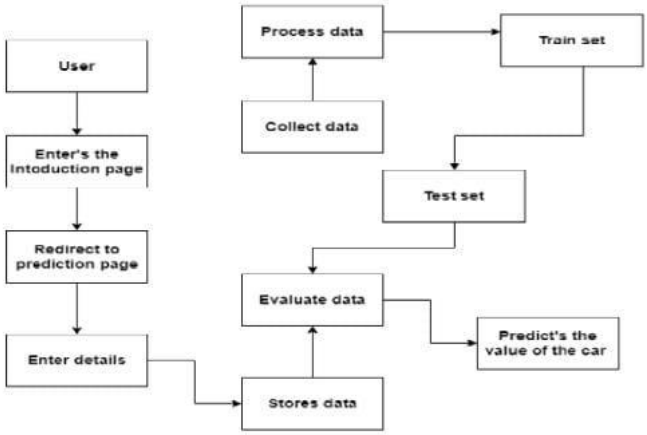
S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Predicting the performance level of cars is an important and interesting problem. The main goal is to predict the performance of the car to improve certain behaviours of the vehicle. This can significantly help to improve the system's fuel consumption and increase efficiency. The performance analysis of the car is based on the engine type, no of engine cylinders, fuel type, horsepower, etc. These are the factors on which the health of the car can be predicted. It is an on-going process of obtaining, researching, analysing, and recording health based on the above three factors. The performance objectives like mileage, dependability, flexibility and cost can be grouped together to play a vital role in the prediction engine and engine management system. This approach is a very important step towards understanding the vehicle's performance.
2.	Idea / Solution description	To train the system with the dataset using a regression model and it will be integrated to the web-based application where the user is notified with the status.

Project Design Phase-II

Data Flow Diagram & User Stories

Date	21 October 2022
Team ID	PNT2022TMID54018
Project Name	Machine Learning based Vehicle Performance Analyzer
Maximum Marks	4 Marks

Data Flow Diagram:



User Stories

UserType	Functional Requirement (Epic)	UserStory Number	User Story/Task	Acceptance criteria	Priority	Release
Customer (web user)	Enters the browser	USN-1	As a user, I canaccess to website using a web browser, the web should contain a simple login page.	I can enter by selecting the appropriate we blink	High	Sprint-1
		USN-2	As a user, I can proceed to the prediction page by selecting the check value button in the home page	I can enter into it without any acceptance	High	Sprint-1
Customer (mobile user)	Enters into a mobile browser	USN-3	As a user, I can use any of the appropriate mobile browser to enter into the website	I can enter by using an appropriate web link	Medium	Sprint-1
Customer Care Executive						
Administrator						
Customer	Access the webpage	USN-4	As a user anyone can access the webpage to check the specifications of the vehicle.	I can access my webpage through online at any time	High	Sprint-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 choose a optimized machine learning algorithm we train most of the regression models to find the most suitable to approach this prediction. Among all the different 4 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.***

- ***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 deployed on IBM cloud and imported using the API key.***

A

44

from sklearn.tree import Decision Tree Regressor

The below code is the model for Decision tree Algorithm.

dt-Decision Tree Regressor(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 below import 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,

A

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('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
111
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 trought request

data = request.get_json(force=True)

prediction = model.v predict(Inp.array(list(data.values())))

output = prediction[0]

return jsonify(output)

if

name

main

":

app.run(debug=True)

TESTING:

TEST CASES:

• 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

RESULTS:

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 are to 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 square error is 6.042499999999999

ADVANTAGES AND DISADVANTAGES:

Advantages:

The model enables an user to immediately analyze a vehicle's performance and provide results instantly.

The model uses decision tree regression which is proved to be more suitable for such cases.

The model takes into account various error factors and acts upon them to produce 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 such as comfort etc.

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 this customers 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 very much 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.

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.

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 overview

of 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 relation across the entire dataframe.

plt.show()

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

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

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 are 95% confident that the correlation between

# the variables is significant.

#

# By convention, when the

# <ul>

#

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

A

# <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'.

A

To

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

#<h5>Conclusion:</h5>

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

#<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-value

of P=", p_value)

 # <h5>Conclusion:</h5>

**# Since the p-value is $<$ 0.1, the correlation between displacement and [5:10 PM, 11/19/2022] ??: mpg 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-value of P=", p_value)

#<h5>Conclusion:</h5>

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

<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-value of P=". P value)

#<h5>Conclusion:</h5>

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

#<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-value of P=" P value)

#<h5>Conclusion:</h5>

Since the p-value is S >S 0.1 , the correlation between acceleration and mpg is statistically significant, but the linear relationship is weak (sim 0.42).

\#< h * 3 > 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-value of P =" p_value)

#<h5>Conclusion: </ h * 5 >

Since the p-value is $<$ 0.001, the correlation between model year and mpg 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-value of P =" , p_value)

\#< h * 5 > Conclusion: </ h * 5 >

\#<> the p-value is S < S 0.001, the correlation between origin and mpg is statistically significant, but the linear relationship is only moderate

(sim 0.563) </ p >

#Ordinary Least Squares 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

#Independent variables

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

dt-Decision Tree Regressor(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 StringIO
from IPython.display import Image from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(dt, out_file-dot_data,
C
A
filled=True, rounded-True,
special_characters=True)
A
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 room for 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 of multiple 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_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)
```

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 this customers 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 very much 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 overview of 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,??
# Visualizations of each attributes w.r.t rest of all attributes
sns.pairplot(dataset,diag_kind='kde') #pairplot represents pairwise relation across the entire dataframe.
plt.show()
# Regression plots(regplot()) creates a regression line between 2 parameters and helps to visualize their linear relationships.
sns.regplot(x="cylinders", y="mpg", data=dataset)
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 are 95% 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-value of P =", p_value)
# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.001, the correlation between cylinders and mpg is statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.
# <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-value of P =", p_value)
# <h5>Conclusion:</h5>
# Since the p-value is $<$ 0.1, the correlation between displacement and mpg 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-value of P =", p_value)
# <h5>Conclusion:</h5>
# Since the p-value is $<$ 0.001, the correlation between horsepower and mpg is statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative and moderately strong.
# <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-value of P =", p_value)
# <h5>Conclusion:</h5>
# 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)
# <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-value of P =", p_value)
# <h5>Conclusion:</h5>
# Since the p-value is $>$ 0.1, the correlation between acceleration and mpg is statistically significant, but the linear relationship is weak (~0.420).
# <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-value of P =", p_value)
# <h5>Conclusion:</h5>
# Since the p-value is $<$ 0.001, the correlation between model year and mpg 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-value of 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
# <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)
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 StringIO
from 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 room for 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 of multiple
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_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 room for 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 the two
distributions overlap a bit. However, there is definitely some room for improvement.
from sklearn.metrics import r2_score,mean_squared_error
r2_score(y_test,y_pred3)
```

```
mean_squared_error(y_test,y_pred3)
np.sqrt(mean_squared_error(y_test,y_pred3))
# <b>Conclusion:</b>
# <p>When comparing models, the model with the higher R-squared value is a better fit for the data.</p>
# <p>When comparing models, the model with the smallest MSE value is a better fit for the data.</p>
#
# Comparing these three models, we conclude that the DecisionTree model is the 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 overview of 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,??

# Visualizations of each attributes w.r.t rest of all attributes
sns.pairplot(dataset,diag_kind='kde') #pairplot represents pairwise relation across the entire dataframe.
plt.show()

# Regression plots(regplot()) creates a regression line between 2 parameters and helps to visualize their linear relationships.
sns.regplot(x="cylinders", y="mpg", data=dataset)
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 are
95% 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-value of P =", p_value)
# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.001, the correlation between cylinders and mpg is statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.
# <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-value of P =", p_value)
# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.1, the correlation between displacement and mpg 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-value of P =", p_value)
# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.001, the correlation between horsepower and mpg is
statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative
and moderately strong.
# <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-value of 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-value of 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-value of P =", p_value)
# <h5>Conclusion:</h5>
# <p>Since the p-value is $<$ 0.001, the correlation between model year and mpg 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-value of 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).</p>
# <b>Ordinary Least Squares</b> 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
# <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_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 StringIO
from 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 room for 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 of multiple 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_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 room for 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 the two
distributions overlap a bit. However, there is definitely some room for improvement.
from sklearn.metrics import r2_score,mean_squared_error
r2_score(y_test,y_pred3)
mean_squared_error(y_test,y_pred3)
np.sqrt(mean_squared_error(y_test,y_pred3))
from ibm_watson_machine_learning import APIClient
wml_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=dtsoftware_spec_uid=wml_client.software_specifications.get_uid_by_name("ru
ntime-22.1-py3.9")
model_props={
wml_client.repository.ModelMetaNames.NAME:model_name,
wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:sof tware_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/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') }}">
<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 (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>
</div>
```

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('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 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 plan for a Tour"
return render_template('index.html', prediction_text='{}'.format(ped))
if _name_ == "_main_":
app.run(debug=True)
```