Car Resale Value prediction

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTERPRENEUSHIP - HX8001

Submitted by

MANOJ KUMAR	110119104028
ABDULLAH	110119104001
AMAL	110119104007
BAZITH ALI	110119104016

TEAM ID (PNT2022TMID23912)

Of

Department of Computer Science Engineering

ABSTRACT

This paper aims to build a model to predict used cars' reasonable prices based on multiple aspects, including vehicle mileage, year of manufacturing, fuel consumption, transmission, road tax, fuel type, and engine size. This model can benefit sellers, buyers, and car manufacturers in the used cars market.Upon completion, it can output a relatively accurate price prediction based on the information that user input. The model building process involvesmachine learning and data science. The dataset used was scraped from listings of used cars. Various regression methods, including linear regression, polynomial regression, support vectorregression, decision tree regression, and random forest regression, were appliedin the researchto achieve the highestaccuracy. Before the actual start of model-building, this project visualized the data to understand the dataset better. The dataset was divided and modified to fit the regression, thus ensure the performance of the regression. To evaluatethe performance of each regression, R- square was calculated. Among all regressions in this project,random forest achievedthe highest R-square of 0.90416. Compared to previousresearch, the resultingmodel includes more aspects of used cars while also having a higherprediction accuracy.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

The main idea of making a car resale value predictionsystem is to get hands-on practice for python using Data Science.

i. Car resalevalue prediction is the system to predict the amount of resale value based on the parameters providedby the user.

ii. User enters the details of the car into the form given and accordingly the car resale value is predicted.

iii. The systemis defined in the pythonlanguage that predictsthe amount of resale value based on the given information.

iv. The system works on the trained dataset of the machine learning program that evaluatesthe precise value of the car.

v. User can enter detailsonly of fields like purchaseprice of car, kilometers driven, fuel of car, year of purchase.

1.2 PURPOSE

vi. This resale value prediction system is made for generalpurpose to just predict the amount that can be roughly acquiredby the user.

vii. We try to predict the amount of resale by best 70% accuracy so the user can get estimated value before he resales the car and doesn't make a deal in loss.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The Problem Is Defined As The Optimised Way To Estimateinsurance Cost Based On The Manufacturer With Some Additional Costs Incurred By The Government In The Form Of Taxes.As The Existing Methods For Estimating The Cost Takes A Lot Of Time And Energy And Due To The Increased Price Of New Cars And The Inability Of Customers To Buy New Cars Due To The Lack Of Funds, Used Carssales Are On A Globalincrease. The Pricesof New Cars In The Industry Is Fixed By The So, Customers Buying A New Car Can Be Assuredof The Money They Investto Be Worthy. There Is A Need For A Used Car Price Prediction System To Effectively Determinethe Worthiness Of The Car Using A Varietyof Features. Even Though There Are Websitesthat Offer This Service,their Prediction Method May Not Be The Best. Besides,different Models And Systemsmay Contribute To Predicting Power For A Used Car's Actual Marketvalue. It Is Important To Know Their Actual Marketvalue While Both Buying And Selling.

2.2 REFERENCES

1.Sameerchand Pudaruth, "Predicting the Price of Used Cars using Machine LearningTechniques";(IJICT 2014)

2.Enis gegic,Becir Isakovic, Dino Keco, Zerina Masetic, JasminKevric, "Car Price Prediction UsingMachine Learning"; (TEM Journal 2019)

3.Shonda Kuiper (2008) Introduction to Multiple Regression: How Much Is Your Car Worth?,Journal of Statistics Education, 16:3.

4.Geurts P. (2009)Bias vs Variance Decomposition for Regression and Classification. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge DiscoveryHandbook. Springer, Boston,MA

5.Ning sun, Hongxi Bai, Yuxia Geng, Huizhu Shi, "Price Evaluation Model In Second Hand Car System Based On BP Neural NetworkTheory"; (Hohai University Changzhou, China)

6.Nitis Monburinon, Prajak Chertchom, ThongchaiKaewkiriya, Suwat Rungpheung, Sabir Buya, Pitchayakit Boonpou, "Prediction of Prices for Used Car by using Regression Models"

2.3 PROBLEM STATEMENT DEFINITION

IdeationPhase

Define theProblem

Statements

Date	27 September 2022
Team ID	PNT2022TMID23912
Project Name	Carresalevalueprediction
Maximum Marks	2 Marks

Customer Problem Statement:

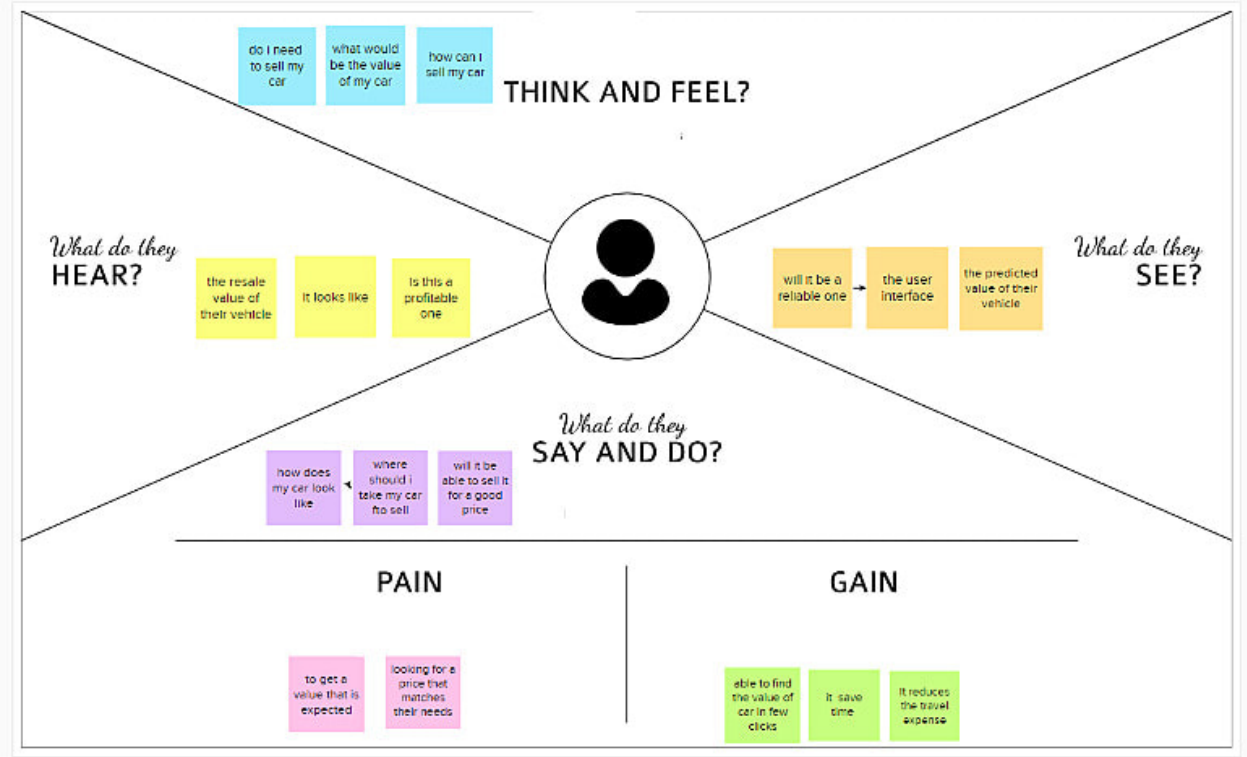
Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Owner of a car	Sell	I don'tknow aboutits current value	Of it wear and tear	Confused
PS-2	Owner of a car	Findits resale value	I am concerned about the minor dents	Of a small accident	A little worried

3.IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map is a simple,easy-to-digest visual that captures knowledgeabout a user's behaviours and attitudes. It is a useful tool to helps teams betterunderstand their users.

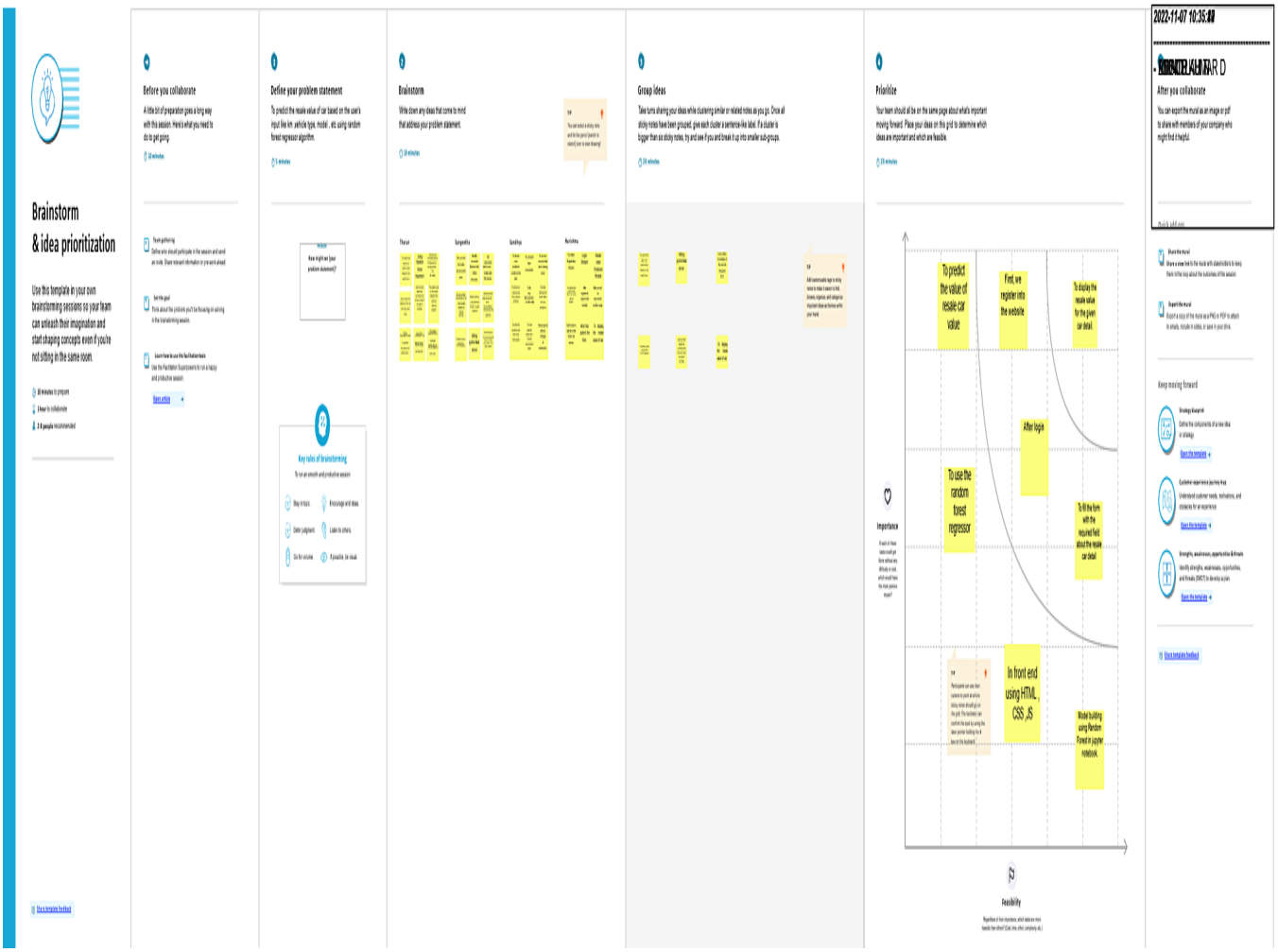
Creating an effectivesolution requires understanding the true problemand the personwho is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 IDEATION AND BRAINSTORMING

Brainstorm:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creativethinking process that leads to problem solving.Prioritizing volume over value, out-of-the-box ideas are welcomeand built upon, and all participants are encouraged to collaborate, helpingeach other developa rich amount of creativesolutions.Use this template in your own brainstorming sessionsso your team can unleashtheir imagination and start shapingconcepts even if you're not sitting in the same room.

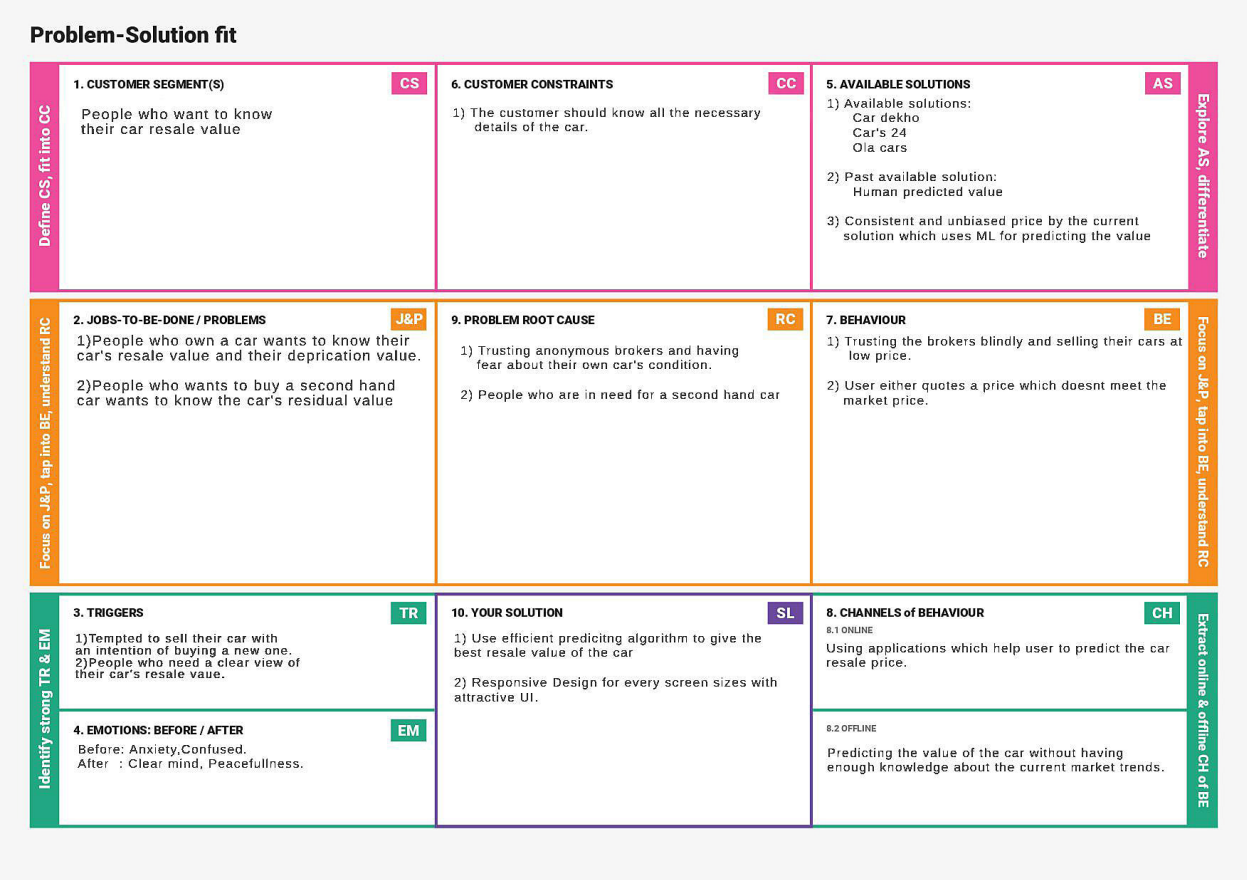


3.3 PROPOSED SOLUTION

S. No.	Parameter	Description
1.	**Problem Statement (Problem to be solved)**	With difficult economic conditions, it is likely thatsales of second-hand imported (reconditioned) carsand used cars will increase. In many developed countries, it is common to lease a car rather than buying it outright. Once the lease period is over, the buyer has the possibility to buy thecar at its residual value,i.e., it's expected resale value. So, we need to predict a car's resalevalue basedon minimal features like mileage, kilometers driven, condition of the car, etc. Thus, it is of commercial interestto sellers/financers to be ableto predict the salvagevalue (residual value)of cars with accuracy.
2.	**Idea / Solution Description**	Our mainidea for predicting a car's resalevalue is to havea dynamic and most fittingalgorithm that analyzes the vehicletype, model, fuel type,kilometers driven, etc.,which generates an approximate marketprice of the car.
3.	**Novelty / Uniqueness**	We generate a detailed reportthat assists the buyer with the best practices to maintain a car and also produces an approximate schedule for the vehicle's maintenance.
4.	**Social Impact / Customer Satisfaction**	Usage of second-handcars reduces the impact on the environment. It also

		speaks about the demand of the cars in

3.4 PROBLEM SOLUTION FIT



REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposedsolution.

FR No.	FunctionalRequirement (Epic)	Sub Requirement(Story / Sub-Task)
FR-1	CarRegistration	Registration throughWebsite
FR-2	CarValue Prediction	Predicting the car resalevalue
FR-3	Suggesting Buyers	Suggesting thebuyerthrough website

4.2 NON-FUNCTIONAL REQUIREMENTS

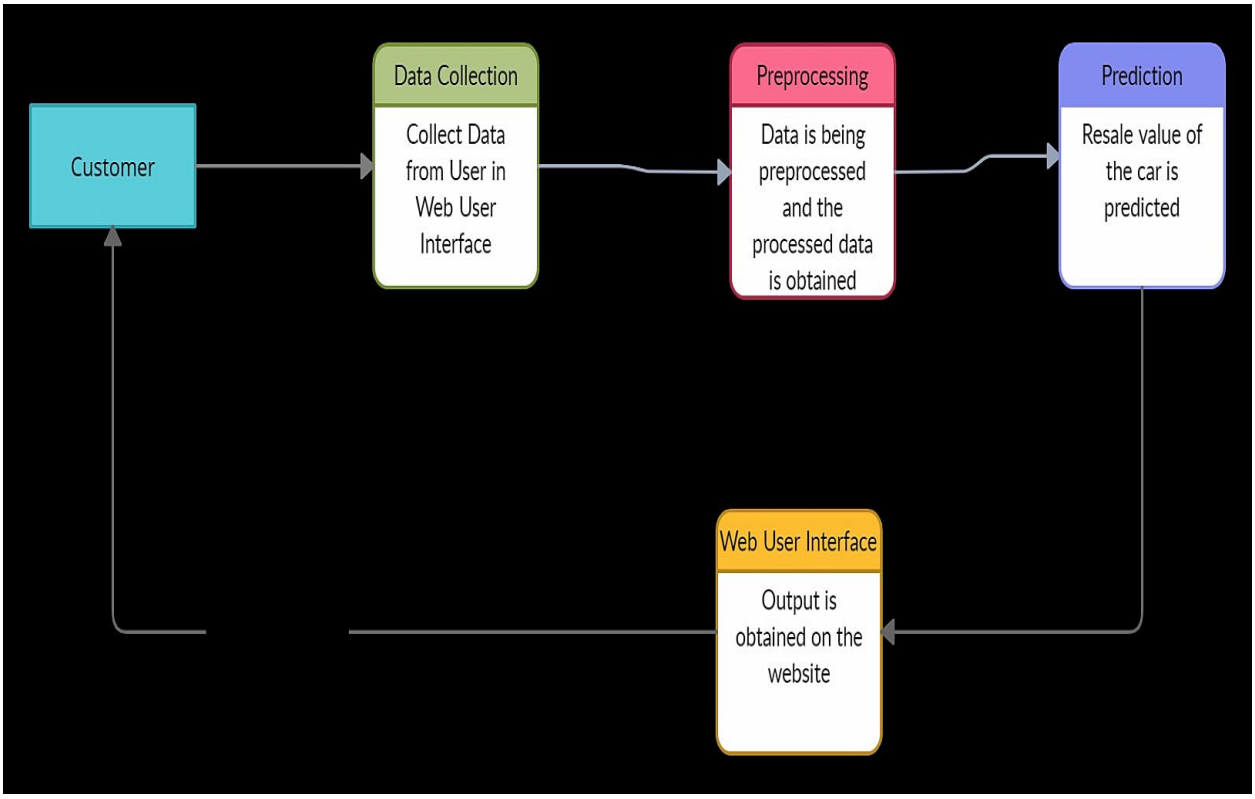
Following are the non-functionalrequirements of the proposedsolution.

FR No.	Non-FunctionalRequirement	Description
NFR-1	**Usability**	Predicting the car'sresale value
NFR-2	**Security**	Providing baselevelsecurity for website
NFR-3	**Reliability**	Providing accurateresale value which is reliable
NFR-4	**Performance**	Enhancing performance by using efficient machine learning algorithm
NFR-5	**Availability**	Availableanytime
NFR-6	**Scalability**	Can handlemanyusers ata time

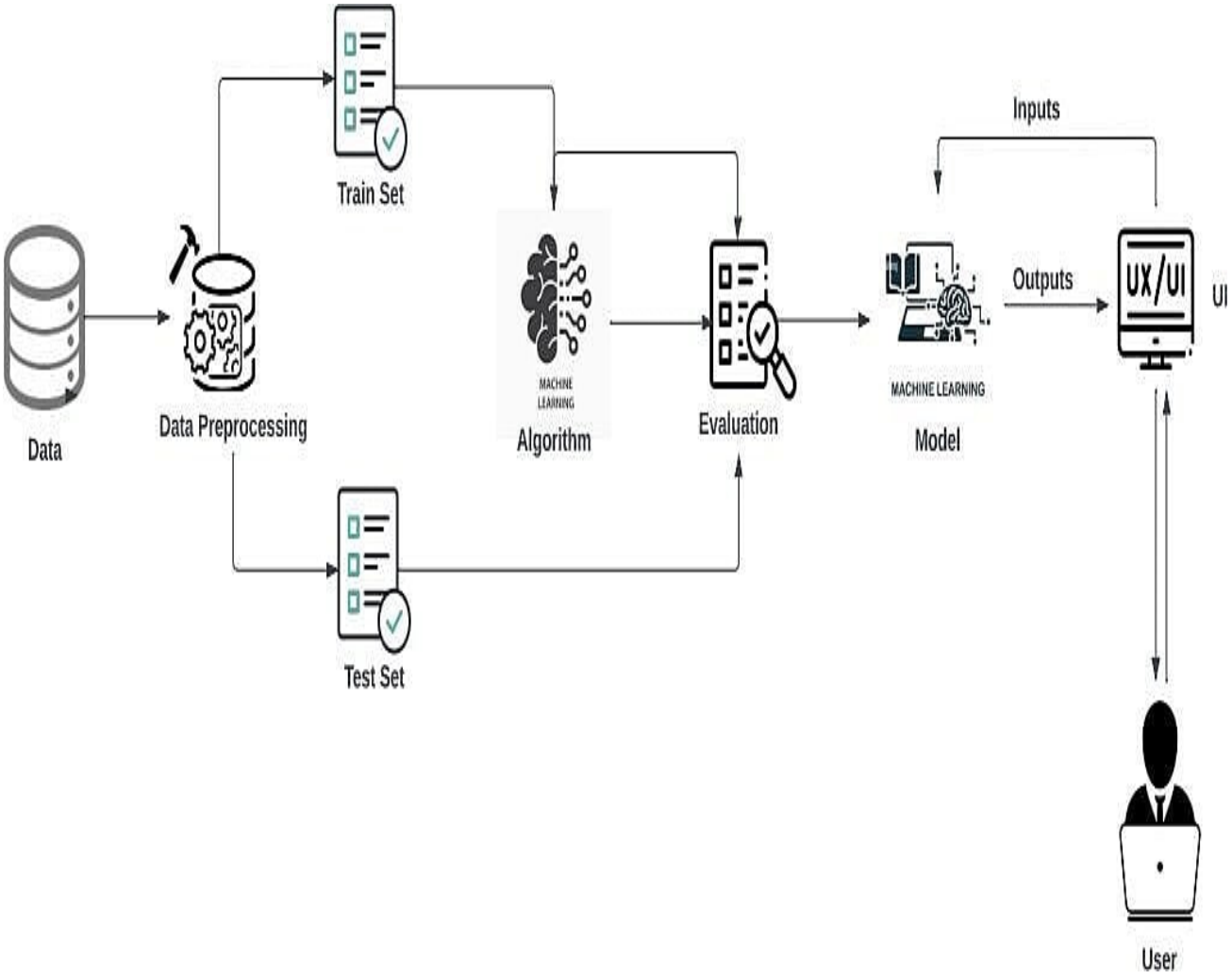
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

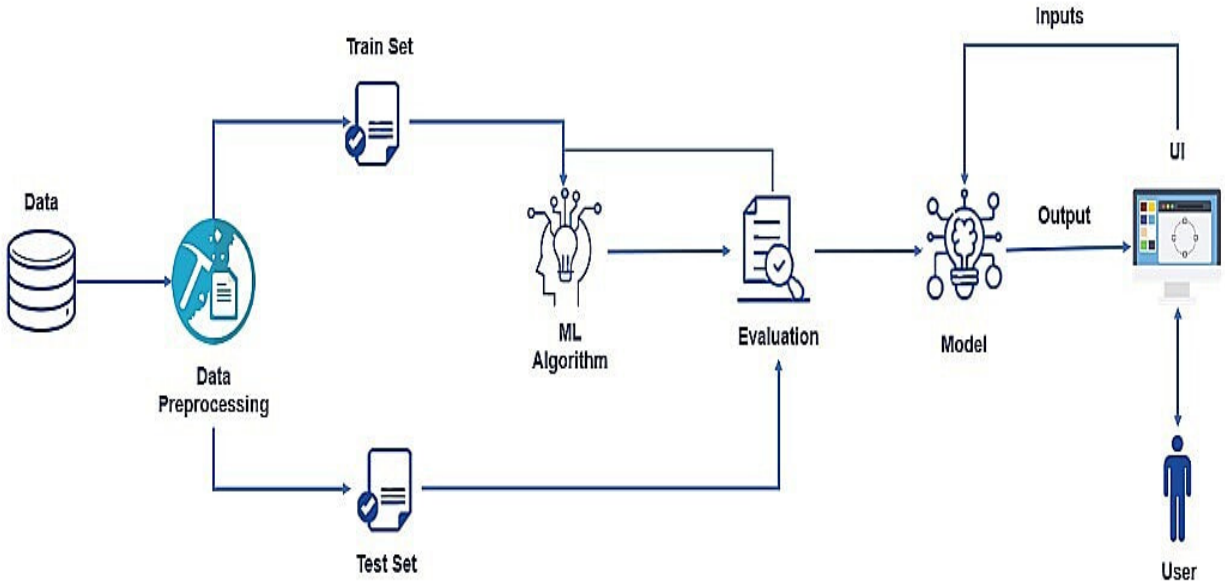
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depictthe right value of the resale car of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2.1 SOLUTION ARCHITECTURE



5.2.2 TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Stories:

UserType	Functional Requirement (Epic)	User Story Number	UserStory/Task	Acceptance Criteria	Priority	Release
Customer (DesktopUser)	Data Entry	USN - 1	User can enter the car details in thewebsite.	Enters the car details	Medium	Sprint-1
Customer (Desktop User)	Obtain Output	USN - 2	User will receive carresale value in the website.	Receives the car resalevalue	High	Sprint-1

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Project PlanningPhase

Project Planning Template
(ProductBacklog, Sprint Planning,Stories,
Story points)

Date	18 October 2022
Team ID	PNT2022TMID23912
Project Name	Project – CarResale Value Prediction
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below templateto create product backlogand sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Dataset Reading and Preprocessing	USN-1	Cleaning the dataset end spliting to dependent and independentvariables	2	High	MANOJ KUMAR D
Sprint-2	Building the Model	USN-2	Choosing the appropriate modelfor building and saving the modelas pickle file	1	High	ABDULLAH T
Sprint-3	Application Building	USN-3	Using flask deploying the ML model	2	Medium	AMAL P
Sprint-4	Train the Model in ibm	USN-4	Finally train the modelonIBM cloud anddeploythe application	2	Medium	BASITH ALI

Project Tracker, Velocity& BurndownChart: (4 Marks)

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

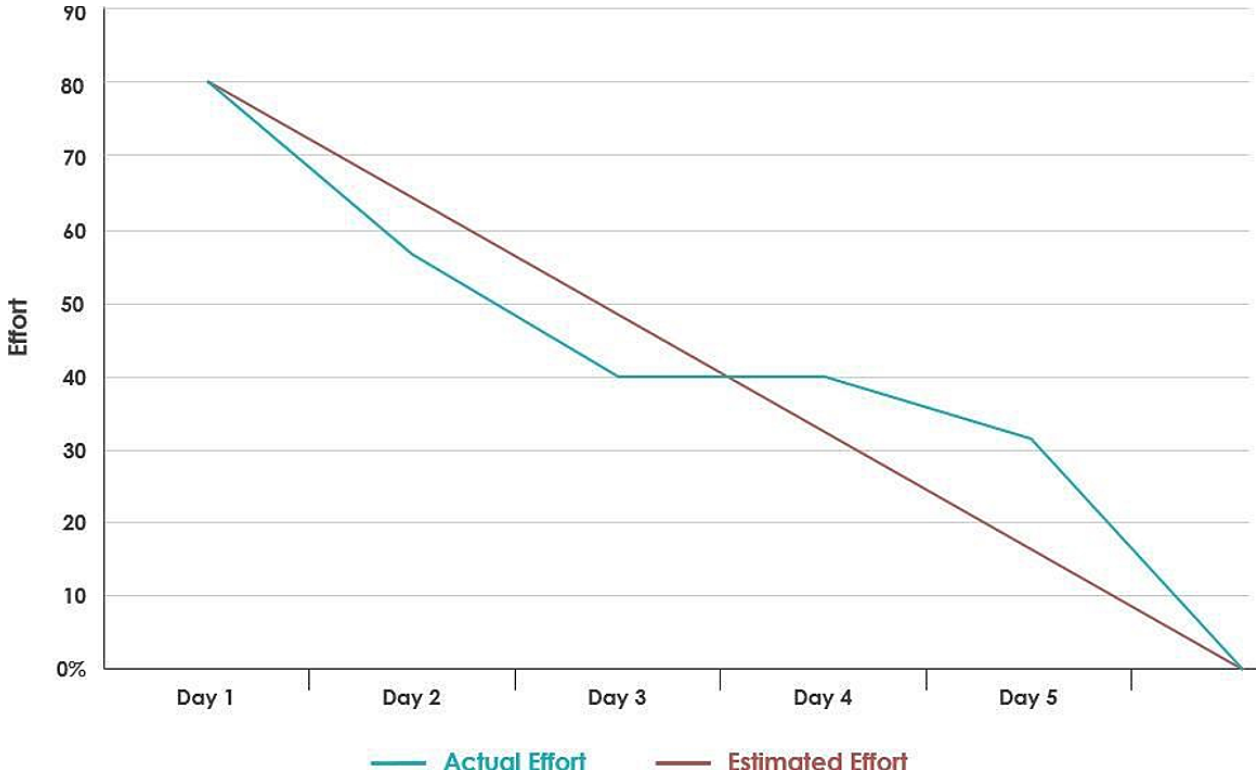
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (pointspersprint). Let's calculatethe team's averagevelocity (AV)periteration unit (story points perday)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{6} = 3.34$$

Burndown Chart:

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



6.2 SPRINT DELIVERY SCHEDULE

SPRINT 1-

1. **Create a basic website that lets user into the website and by using machine learning algorithms that will predict the user details, and provides output.**

2. **Create the Jupyter Notebook with the prediction model algorithm as**

selected.

3. **Using the flask integrate the frontend and backend.**

SPRINT 2-

1. **Enhance the feature of website**
2. **Tunn the Notebook to better predict by hypertunning the parameters**

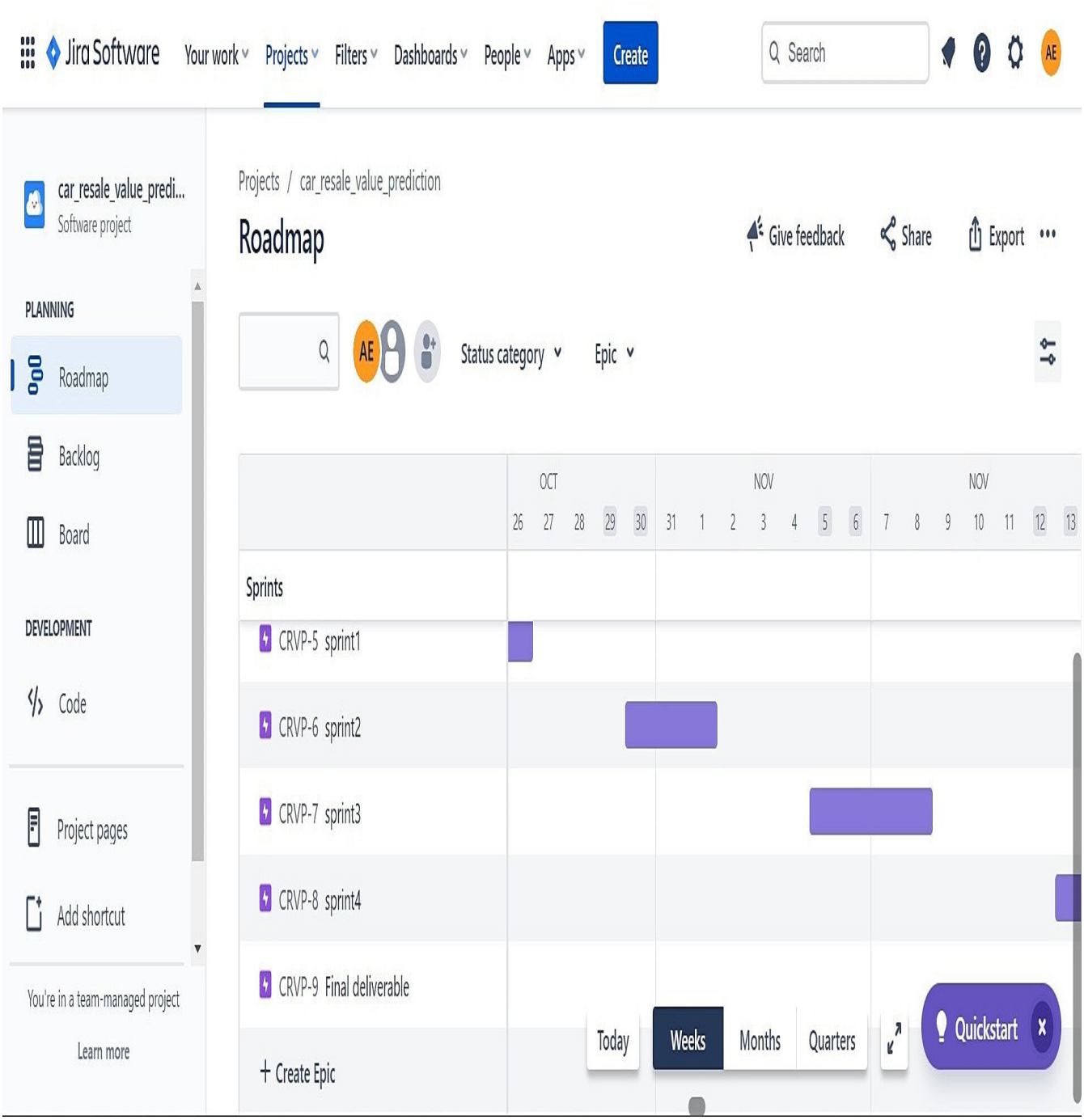
SPRINT 3-

1. **Enhance the website features**
2. **Jupyter Notebook**

SPRINT 4-

1. **Various car models included.**
2. **Prediction tool that gives visual results and the approximate value is predicted.**
3. **The predicted value will appear in fraction of seconds so the user could see the prediction instantly.**

6.3 REPORTS FROM JIRA



7.CODING AND SOLUTIONING

7.1 FEATURE 1(FLASK APP)

app.py

```
from flask import Flask, render_template, request, redirect
from flask_cors import CORS, cross_origin
import pickle
import pandas as pd
import numpy as np


# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "8MK3A0eyRIQXgHpK0FjRMhfTXdaRMOhiGHVBmySZrQMh"
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}



app = Flask(name)
cors = CORS(app)


model = pickle.load(open('LinearRegressionModel.pkl', 'rb'))
car = pd.read_csv('Autos_Cleaned_Data.csv')



@app.route('/', methods=['GET', 'POST'])
def index():
    companies = sorted(car['company'].unique())
    car_models = sorted(car['name'].unique())
    vehicle_type = sorted(car['vehicle_type'].unique())
    year = sorted(car['year'].unique(), reverse=True)
    month = sorted(car['month'].unique())
    fuel_type = sorted(car['fuel_type'].unique())
    gear = sorted(car['gear'].unique())
    damaged = sorted(car['damaged'].unique())
```

```
    companies.insert(0, 'Select Company')
    vehicle_type.insert(0, 'Select Vehicle Type')
    fuel_type.insert(0, 'Select Fuel Type')
    year.insert(0, 'Select Year of Reg')
    month.insert(0, 'Select Month of Reg')
    gear.insert(0, 'Select the Gear Type')
    damaged.insert(0, 'Condition of the Car')



    return render_template('index.html', companies=companies, car_models=car_models, years=year,
                           vehicle_types=vehicle_type, months=month, fuel_types=fuel_type, gears=gear,
damages=damaged)



@app.route('/predict', methods=['POST'])
@cross_origin()
def predict():
    company = request.form.get('company')
    car_model = request.form.get('car_models')
    vehicle_type = request.form.get('vehicle_type')
    year = request.form.get('year')
    month = request.form.get('month')
    gear = request.form.get('gear')
    fuel_type = request.form.get('fuel_type')
    power = request.form.get('ps')
    kms_driven = int(request.form.get('kilo_driven'))
    damaged = request.form.get('damaged')
    print(company,car_model, vehicle_type, year, month, gear, fuel_type, power, kms_driven, damaged )



    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"fields": ['company', 'name', 'vehicle_type', 'year', 'month', 'gear',
'fuel_type', 'ps', 'kms_driven', 'damaged'], "values": [company,car_model, vehicle_type, year, month, gear,
fuel_type, power, kms_driven, damaged]}]}


    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/0e796fd8-
2ea1-43f3-87a9-6b1ca049389f/predictions?version=2022-11-19', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
  predictions = response_scoring.json()
```

```
print(response_scoring.json())
prediction = predictions['predictions'][0]['values'][0][0]
print(prediction)



    return str(np.round(prediction[0], 2))



if name == 'main':
    app.run()
```

7.2 FEATURE 2 (JUPITER FILE)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
mpl.style.use('ggplot')
```

In [2]:
```
car=pd.read_csv('Autos_Data.csv')
```

In [3]:
```
car.head()
```

Out[3]:

	Unnamed: 0	company	name	vehicle_type	year	month	gear	fuel_type	ps	kms_driv
0	1	Opel	Opel Insignia 2.8	Kombi	2012	Dec	Automatic	Petrol	325	600
1	2	Opel	Opel Insignia 2.8	Kombi	2011	Sep	Automatic	Petrol	325	900
2	3	Opel	Opel Insignia 2.8	Kombi	2009	May	Manual	Petrol	260	1000
3	4	Opel	Opel Insignia 2.8	Kombi	2010	Jun	Manual	Petrol	325	1000
4	5	Opel	Opel Insignia 2.8	Kombi	2009	May	Automatic	Petrol	260	1250

In [4]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56681 entries, 0 to 56680
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    56681 non-null  int64
 1   company       56681 non-null  object
 2   name          56681 non-null  object
 3   vehicle_type  56681 non-null  object
 4   year          56681 non-null  int64
 5   month         56681 non-null  object
 6   gear          56681 non-null  object
 7   fuel_type     56681 non-null  object
 8   ps            56681 non-null  int64
 9   kms_driven    56681 non-null  int64
 10  damaged       56681 non-null  object
 11  price         56681 non-null  int64
dtypes: int64(5), object(7)
memory usage: 5.2+ MB
```

In [5]:
```
car['price']=car['price'].astype(int)
```

In [6]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56681 entries, 0 to 56680
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    56681 non-null  int64
 1   company       56681 non-null  object
 2   name          56681 non-null  object
 3   vehicle_type  56681 non-null  object
 4   year          56681 non-null  int64
 5   month         56681 non-null  object
 6   gear          56681 non-null  object
 7   fuel_type     56681 non-null  object
 8   ps            56681 non-null  int64
 9   kms_driven    56681 non-null  int64
 10  damaged       56681 non-null  object
 11  price         56681 non-null  int32
dtypes: int32(1), int64(4), object(7)
memory usage: 5.0+ MB
```

In [7]:
```
car['name']=car['name'].str.replace('_','-')
```

In [8]:
```
car['name']
```

Out[8]:
```
0        Opel Insignia 2.8
1        Opel Insignia 2.8
2        Opel Insignia 2.8
3        Opel Insignia 2.8
4        Opel Insignia 2.8
              ...
56676      Mercedes Benz C
56677      Mercedes Benz C
56678      Mercedes Benz C
56679      Mercedes Benz C
56680      Mercedes Benz C
Name: name, Length: 56681, dtype: object
```

In [9]:
```
car['name']=car['name'].str.split('-').str.slice(0,3)
```

In [10]:
```
car['name']=car['name'].str.join(' ')
```

In [11]:
```
car['name']
```

Out[11]:
```
0        Opel Insignia 2.8
1        Opel Insignia 2.8
2        Opel Insignia 2.8
3        Opel Insignia 2.8
4        Opel Insignia 2.8
              ...
56676      Mercedes Benz C
56677      Mercedes Benz C
56678      Mercedes Benz C
56679      Mercedes Benz C
56680      Mercedes Benz C
Name: name, Length: 56681, dtype: object
```

In [13]:
```
car=car.drop('Unnamed: 0', axis=1)
```

In [14]:
```
car
```

Out[14]:

	company	name	vehicle_type	year	month	gear	fuel_type	ps	kms_driven	d
0	Audi	Audi A3 1.4	Kombi	2014	Apr	Manual	CNG	110	30000	
1	Audi	Audi A3 G	Kombi	2014	Jun	Manual	CNG	110	30000	
2	BMW	BMW 316I 316G	Limousine	2000	Apr	Manual	CNG	102	125000	
3	BMW	BMW 316G compact	Limousine	2000	Apr	Manual	CNG	102	150000	
4	BMW	BMW E60 2010R	Limousine	2010	Feb	Manual	CNG	177	150000	
...	...	...	...	...	...	...	...	...	...	...
187489	Volvo	Volvo XC90 V8	SUV	2005	Apr	Automatic	Petrol	315	150000	
187490	Volvo	Volvo XC90 3.2	SUV	2007	Nov	Automatic	Petrol	238	150000	
187491	Volvo	Volvo XC90 V8	SUV	2007	Jun	Automatic	Petrol	315	150000	
187492	Volvo	Volvo XC90 V8	SUV	2008	Apr	Automatic	Petrol	315	150000	
187493	Volvo	Volvo XC90 2.5T	SUV	2004	Sep	Manual	Petrol	209	150000	

187494 rows × 11 columns

In [14]:
```
car.to_csv('Autos_Cleaned_Data.csv')
```

In [15]:
```
car.dtypes
```

Out[15]:
```
company         object
name            object
vehicle_type    object
year             int64
month           object
gear            object
fuel_type       object
ps               int64
kms_driven       int64
damaged         object
price            int32
dtype: object
```

In [16]:
```
car
```

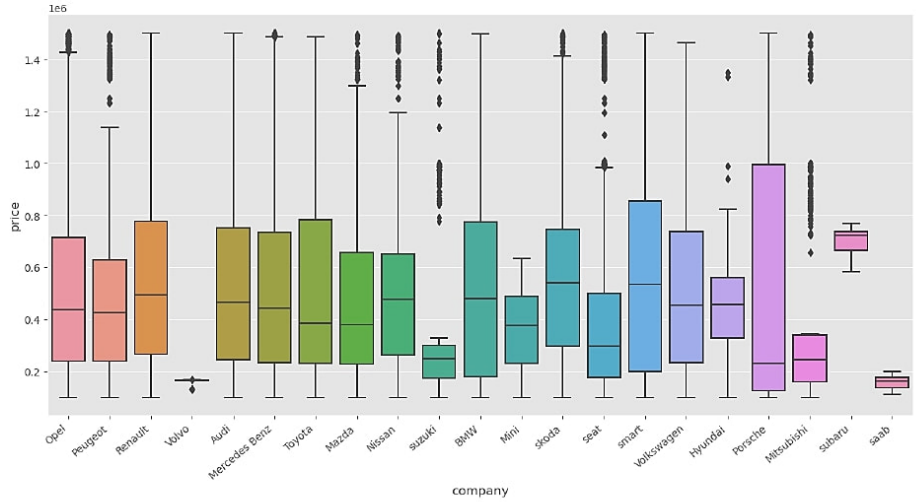
Out[16]:

	company	name	vehicle_type	year	month	gear	fuel_type	ps	kms_driven	d
0	Opel	Opel Insignia 2.8	Kombi	2012	Dec	Automatic	Petrol	325	60000	
1	Opel	Opel Insignia 2.8	Kombi	2011	Sep	Automatic	Petrol	325	90000	
2	Opel	Opel Insignia 2.8	Kombi	2009	May	Manual	Petrol	260	100000	
3	Opel	Opel Insignia 2.8	Kombi	2010	Jun	Manual	Petrol	325	100000	
4	Opel	Opel Insignia 2.8	Kombi	2009	May	Automatic	Petrol	260	125000	
...	...	...	...	...	...	...	...	...	...	...
56676	Mercedes Benz	Mercedes Benz C	Limousine	2008	Jul	Manual	Petrol	184	150000	
56677	Mercedes Benz	Mercedes Benz C	Limousine	2009	Apr	Manual	Petrol	184	150000	
56678	Mercedes Benz	Mercedes Benz C	Limousine	2009	Jul	Manual	Petrol	231	150000	
56679	Mercedes Benz	Mercedes Benz C	Limousine	2010	Mar	Manual	Petrol	184	150000	
56680	Mercedes Benz	Mercedes Benz C	Limousine	2010	Mar	Manual	Petrol	184	150000	

56681 rows × 11 columns

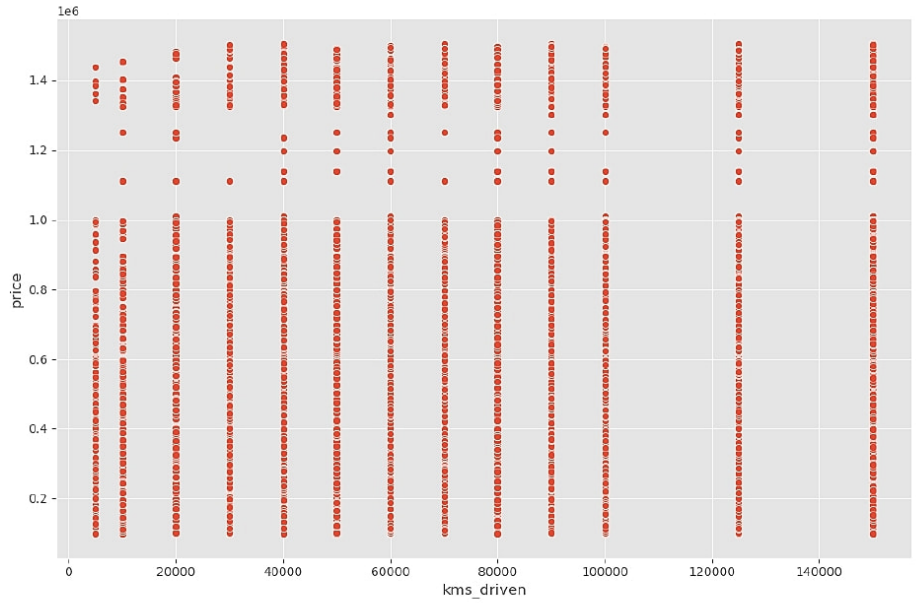
In [18]:
```
import seaborn as sns
```

In [19]:
```
plt.subplots(figsize=(15,7))
ax=sns.boxplot(x='company',y='price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```



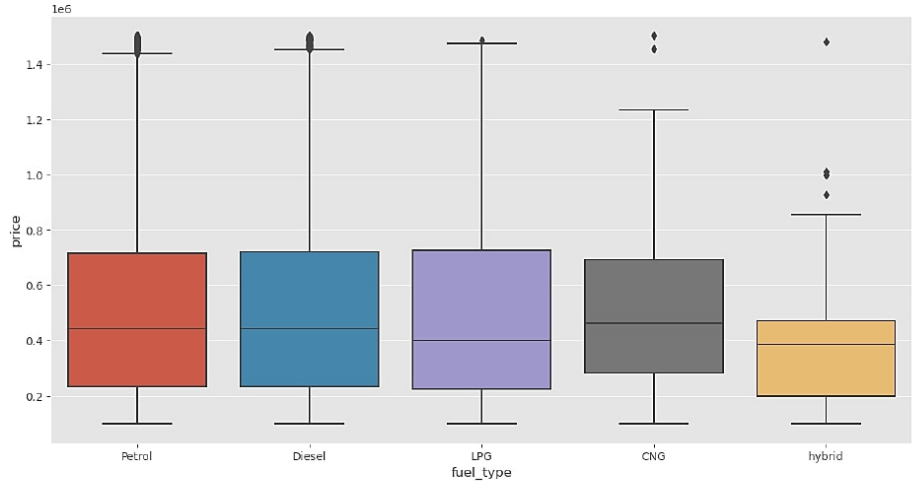
In [22]:
```
sns.relplot(x='kms_driven',y='price',data=car,height=7,aspect=1.5)
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x21037b175b0>
```



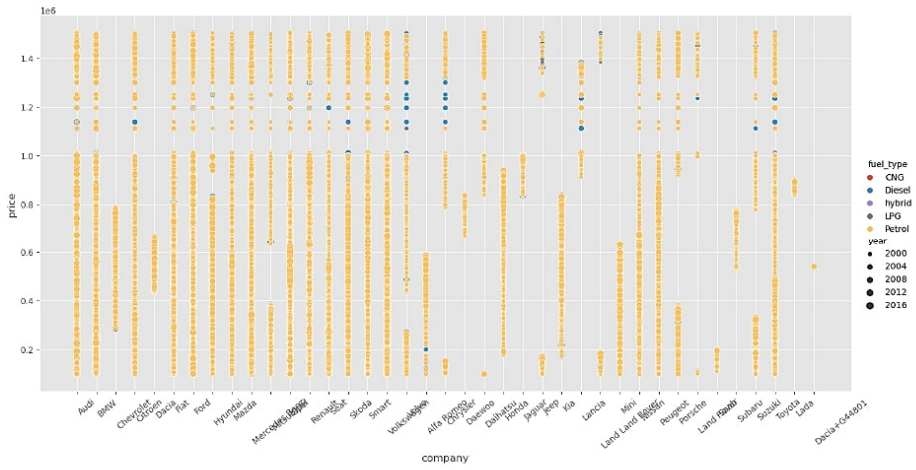
In [23]:
```
plt.subplots(figsize=(14,7))
sns.boxplot(x='fuel_type',y='price',data=car)
```

Out[23]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='price'>
```



In [23]:
```
ax=sns.relplot(x='company',y='price',data=car,hue='fuel_type',size='year',height=7,
ax.set_xticklabels(rotation=40,ha='left')
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x2a5b0c851f0>
```



Extracting Training Data

In [24]:
```
x=car[['company','name','vehicle_type','year', 'month', 'gear', 'fuel_type', 'ps',
y=car['price']
```

In [25]:
```
x
```

Out[25]:

	company	name	vehicle_type	year	month	gear	fuel_type	ps	kms_driven	d
0	Opel	Opel Insignia 2.8	Kombi	2012	Dec	Automatic	Petrol	325	60000	
1	Opel	Opel Insignia 2.8	Kombi	2011	Sep	Automatic	Petrol	325	90000	
2	Opel	Opel Insignia 2.8	Kombi	2009	May	Manual	Petrol	260	100000	
3	Opel	Opel Insignia 2.8	Kombi	2010	Jun	Manual	Petrol	325	100000	
4	Opel	Opel Insignia 2.8	Kombi	2009	May	Automatic	Petrol	260	125000	
...	...	...	...	...	...	...	...	...	...	...
56676	Mercedes Benz	Mercedes Benz C	Limousine	2008	Jul	Manual	Petrol	184	150000	
56677	Mercedes Benz	Mercedes Benz C	Limousine	2009	Apr	Manual	Petrol	184	150000	
56678	Mercedes Benz	Mercedes Benz C	Limousine	2009	Jul	Manual	Petrol	231	150000	
56679	Mercedes Benz	Mercedes Benz C	Limousine	2010	Mar	Manual	Petrol	184	150000	
56680	Mercedes Benz	Mercedes Benz C	Limousine	2010	Mar	Manual	Petrol	184	150000	

56681 rows × 10 columns

In [28]:
```
y.shape
```

Out[28]:
```
(56681,)
```

Applying Train Test Split

In [29]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

Creating an OneHotEncoder object to contain all the possible categories

In [32]:
```
ohe=OneHotEncoder()
ohe.fit(x[['company','name','vehicle_type','month', 'gear', 'fuel_type', 'damaged'
```

Out[32]:
```
OneHotEncoder()
```

Creating a column transformer to transform categorical columns

In [33]:
```
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['
                                     remainder='passthrough')
```

Linear Regression Model

In [34]:
```
lr=LinearRegression()
```

Making a pipeline

In [35]:
```
pipe=make_pipeline(column_trans,lr)
```

Fitting the model

In [36]:
```
pipe.fit(x_train,y_train)
```

Out[36]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(categories=[array
(['Audi', 'BMW', 'Hyundai', 'Mazda', 'Mercedes Benz', 'Mini',
       'Mitsubishi', 'Nissan', 'Opel', 'Peugeot', 'Porsche', 'Renault',
       'Toyota', 'Volkswagen', 'Volvo', 'saab', 'seat', 'skoda', 'smart',
       'subaru', 'suzuki'], dtype=object),
                                                                             ar...
                                                                             array
(['Apr', 'Aug', 'Dec', 'Feb', 'Jan', 'Jul', 'Jun', 'Mar', 'May',
       'Nov', 'Sep'], dtype=object),
                                                                             array
(['Automatic', 'Manual'], dtype=object),
                                                                             array
(['CNG', 'Diesel', 'LPG', 'Petrol', 'hybrid'], dtype=object),
                                                                             array
(['No', 'Not Declared', 'Yes'], dtype=object)]),
                                                  ['company', 'name',
                                                   'vehicle_type', 'month',
                                                   'gear', 'fuel_type',
                                                   'damaged'])])),
                ('linearregression', LinearRegression())])
```

In [37]:
```
y_pred=pipe.predict(x_test)
```

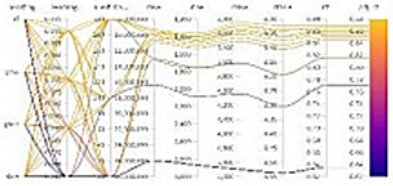
Checking R2 Score

In [38]:
```
r2_score(y_test,y_pred)
```

Out[38]:
```
0.002618508194348834
```

8.TESTING

TEST CASES:

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** LGBM Regressor MAE: 1327.56 MSE: 9492244.25 RMSE: 3080.93 RMSLE: 8.05 R2 Score: 0.8664 Adjusted R2 Score: 0.8666	
2.	Tune the Model	**Hyperparameter Tuning** 1) Learning Rate: [0.01, 0.03, 0.05, 0.07] 2) Boosting Type: ['gbdt','dart','goss','rf'] 3) Number of Estimators: [100,200,300] **Validation Method**: Grid Search Cross Validation **Best Parameters:** Learning Rate – 0.07 Boosting Type – 'gbdt' Number of Estimators - 300	 

8.2 USER ACCEPTANCE TESTING:

				Date	17-Nov-22								
				Team ID	PNT2022TMID40127								
				Project Name	Project - Car Resale Value Prediction								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
HomePage_TC_001	UI	Home Page	Verify all the UI elements in Home page rendered properly		1. Enter URL and click go 2. Verify all the UI elements displayed or not	-	All the UI elements rendered properly	Working as expected	Pass		N		Ajith
HomePage_TC_002	Functional	Home Page	Verifiy the Data Entry page can be reachable.		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button.	-	User should navigate to Data Entry Page	Working as expected	Pass		N		Navinvarma
DataEntryPage_TC_001	UI	Data Entry Page	Verify all the UI elements in Data Entry page rendered properly		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button in the home page 4. Verify all the UI elements displayed or not	-	All the UI elements rendered properly	Working as expected	Pass		N		Rahul
DataEntryPage_TC_002	Functional	Data Entry Page	Verify user is able to enter all values		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button in the home page 4. Verify all the UI elements displayed or not 5. Verify if all values can be entered	2012 12 12 12 Manual Yes Golf Volkswagen Petrol Coupe	User should be able to enter all values in data entry page	Working as expected	Pass		N		Jothi murugan
DataEntryPage_TC_003	Functional	Data Entry Page	Verifiy the Output Display page can be reachable.		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button in the home page 4. Verify all the UI elements displayed or not 5. Verify if all values can be entered 6. Press the submit Button	-	User should navigate to Output Display Page	Working as expected	Pass		N		Rahul
OutputDisplayPage_TC_001	UI	Output Display Page	Verify all the UI elements in Output Display page rendered properly		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button in the home page 4. Verify all the UI elements displayed or not 5. Verify if all values can be entered 6. Press the submit Button 7. Verify all the UI elements displayed or not	-	All the UI elements rendered properly	Working as expected	Pass		N		Navinvarma
OutputDisplayPage_TC_002	Functional	Output Display Page	Verify user is able to get predicted result		1. Enter URL and click go 2. Verify all the UI elements displayed or not. 3. Press the Check Price button in the home page 4. Verify all the UI elements displayed or not 5. Verify if all values can be entered 6. Press the submit Button 7. Verify all the UI elements displayed or not 8. Verify if the predicted value is displayed or not	-	Predited Car Resale Value is displayed on the page	Working as expected	Pass		N		Ajith

Test Scenarios :

Verify user is able to see home page?
Verify user is able to navigate to data entry page?
Verify user is able to see data entry page?
Verify user is able to enter values in the fields?
Verify user is able to navigate to output display page?
Verify user is able to view the output display page?

9.RESULTS

PERFORMANCE METRICS:

Project Planning Template
(ProductBacklog, Sprint Planning,Stories,
Story points)

Date	18 October 2022
Team ID	PNT2022TMID23912
Project Name	Project – CarResale Value Prediction
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below templateto create product backlogand sprint schedule

Sprint	Functional Requirement (Epic)	User Story Numb er	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Dataset Reading and Preprocessing	USN-1	Cleaning the dataset end spliting to dependent and independentvariables	2	High	MANOJ KUMAR D
Sprint-2	Building the Model	USN-2	Choosing the appropriate modelfor building and saving the modelas pickle file	1	High	ABDULLAH T
Sprint-3	Application Building	USN-3	Using flask deploying the ML model	2	Medium	AMAL P
Sprint-4	Train the Model in ibm	USN-4	Finally train the modelonIBM cloud anddeploythe application	2	Medium	BASITH ALI

10. ADVANTAGES AND DISDVANTAGES

ADVANTAGES

Value for money. Pre-ownedcars come with a lower price tag and offer a much better value for the amountpaid and some other advantages include:

i. Slow rate of depreciation.
ii. Lower insurance and registration charges.
iii. Higher inflation.
iv. Lower loan amount to be borrowed.

DISDVANTAGES

i. Little to No Warranty
ii. New models not avaliable
iii. Little to No Financing
iv. No accurate prediction

11.CONCLUSION

Car price prediction can be a challenging task due to the high number of attributes that should be considered for the accurateprediction. Data cleaningis one of the processesthat increases prediction performance, yet insufficient for the cases of complexdata sets as the one in this research. Applying single machinealgorithm on the data set accuracy was less than 50%. Therefore, the ensemble of multiple machinelearning algorithms has been proposedand this combinationof ML methods gains approximate price prediction.

This is significant improvement compared to single machine learning method approach. However, the drawback of the proposedsystem is that it consumesmuch more computational resources than singlemachine learning algorithm.

12.FUTURE SCOPE

Currently, system can only deal with Swift Dzire cars due to lack of data. Also, data has been collected of only 5 cities of India. This can be extended to multiple car models and cities so as to improveaccuracy and usability.

Efficient use of deep learning such as LSTM (Long shorttermmemory) or RNN (Recurrent Neural networks) can be implemented once enough data is collected. This can improve accuracy and decrease RMSE drastically.One can also implementCNN to determine physical condition of the car from images like identifying dents, scratches etc. and thus predicting more relevant resale value of a car.

13.APPENDIX CODING

Source code:

style.css

```
/* ===== Google Font Import - Poppins ===== */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600&display=swap');
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
body{
    min-height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background: #4070f4;
}
```

```
.container{
    position: relative;
    max-width: 1200px;
    width: 100%;
    border-radius: 6px;
    padding: 30px;
    margin: 0 15px;
    background-color: #fff;
    box-shadow: 0 5px 10px rgba(0,0,0,0.1);
}
.container header{
    position: relative;
    font-size: 20px;
    font-weight: 600;
    color: #333;
}

.container header::before{
    content: "";
    position: absolute;
    left: 0;
    bottom: -2px;
    height: 3px;
    width: 27px;
    border-radius: 8px;
    background-color: #4070f4;
}
.container form{
    position: relative;
    margin-top: 16px;
    min-height: 490px;
    background-color: #fff;
    overflow: hidden;
}

.container form .fields{
    display: flex;
```

```
    align-items: center;
    justify-content: space-between;
    flex-wrap: wrap;
}
form .fields .input-field{
    display: flex;
    width: calc(100% / 3 - 15px);
    flex-direction: column;
    margin: 4px 0;
}
.input-field label{
    font-size: 12px;
    font-weight: 500;
    color: #2e2e2e;
}
.input-field input, select{
    outline: none;
    font-size: 14px;
    font-weight: 400;
    color: #333;
    border-radius: 5px;
    border: 1px solid #aaa;
    padding: 0 15px;
    height: 42px;
    margin: 8px 0;
}
.input-field input :focus,
.input-field select:focus{
    box-shadow: 0 3px 6px rgba(0,0,0,0.13);
}
.input-field select,

.input-field input[type="number"]{
    color: #707070;
}
.input-field input[type="number"]:valid{
    color: #333;
```

```
}
.container form button{
    display: flex;
    align-items: center;
    justify-content: center;
    height: 45px;
    max-width: 200px;
    width: 100%;
    border: none;
    outline: none;
    color: #fff;
    border-radius: 5px;
    margin: 25px 0;
    background-color: #4070f4;
    transition: all 0.3s linear;
    cursor: pointer;
}

.container form .btnText{
    font-size: 14px;
    font-weight: 400;
}

form button:hover{
    background-color: #265df2;
}

form .button {
    margin-right: 14px;
}

@media (max-width: 750px) {
    .container form{
        overflow-y: scroll;
    }
    .container form::-webkit-scrollbar{
       display: none;
```

```
    }
    form .fields .input-field{
        width: calc(100% / 2 - 15px);
    }
}

@media (max-width: 550px) {
    form .fields .input-field{
        width: 100%;
    }
}
```

index.html

```
<!DOCTYPE html>
<html lang="en<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

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

    <!----===== Iconscout CSS ===== -->
    <link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

    <!--<title>Responsive Regisration Form </title>-->
</head>

<body>
    <div class="container">
        <header>GET THE ACCURATE RESALE VALUE OF YOUR CAR</header>

        <form method="post" accept-charset="utf-8" name="Modelform">

            <div class="form first">
                <div class="details personal">
                    <span class="title">Car Features</span>
```

```
                <div class="fields">

                    <div class="input-field">
                        <label>Select the Brand:</label>
                        <select class="selectpicker form-control" id="company" name="company" required="1"
onchange="load_car_models(this.id,'car_models')">
                            {% for company in companies %}
                            <option value="{{ company }}">{{ company }}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="input-field">
                        <label>Select the Model:</label>
                        <select class="selectpicker form-control" id="car_models" name="car_models"
required="1"> </select>
                    </div>

                    <div class="input-field">
                        <label>Select the VehicleType:</label>
                        <select class="selectpicker form-control" id="vehicle_type" name="vehicle_type"
required="1">
                            {% for vehicle_type in vehicle_types %}
                            <option value="{{ vehicle_type }}">{{ vehicle_type }}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="input-field">
                        <label>Select the Gear Type:</label>
                        <select class="selectpicker form-control" id="gear" name="gear" required="1">
                            {% for gear in gears %}
                            <option value="{{ gear }}">{{ gear }}</option>
                            {% endfor %}
                        </select>
                    </div>
```

```
                    <div class="input-field">
                        <label>Select the Fuel Type:</label>
                        <select class="selectpicker form-control" id="fuel_type" name="fuel_type" required="1">
                            {% for fuel in fuel_types %}
                            <option value="{{ fuel }}">{{ fuel }}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="input-field">
                        <label>Power of Car in PS:</label>
                        <input type="number" class="form-control" id="ps" name="ps" placeholder="Power of Car
limit under 2000">
                    </div>
                </div>
            </div>

            <div class="details ID">
                <span class="title">Usage Details</span>

                <div class="fields">
                    <div class="input-field">
                        <label>Kilometres Driven</label>
                        <input type="number" class="form-control" id="kilo_driven" name="kilo_driven"
placeholder="Kms Driven under limit under 2 lakh">
                    </div>
<div class="input-field">
                        <label>Damaged or Repaired</label>
                        <select class="selectpicker form-control" id="damaged" name="damaged" required="1">
                            {% for damaged in damages %}
                            <option value="{{ damaged }}">{{ damaged }}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="input-field">
```

```
                    <label>Manufactured Month</label>
                    <select class="selectpicker form-control" id="month" name="month" required="1">
                        {% for month in months %}
                        <option value="{{ month }}">{{ month }}</option>
                        {% endfor %}
                    </select>
                </div>

                <div class="input-field">
                    <label>Manufactured Year</label>
                    <select class="selectpicker form-control" id="year" name="year" required="1">
                        {% for year in years %}
                        <option value="{{ year }}">{{ year }}</option>
                        {% endfor %}
                    </select>
                </div>
            </div>

            <button  class="btn btn-primary form-control" onclick="send_data()" >PREDICT</button>
            <div>

            </div>
        </div>
    </div>
</form>
    <div class="col-12" style="text-align: center">
        <h1 style="color:#008000"><span id="prediction"></span></h1>
    </div>
</div>


<script>
const form = document.querySelector("form"),
      nextBtn = form.querySelector(".nextBtn"),
      backBtn = form.querySelector(".backBtn"),
      allInput = form.querySelectorAll(".first input");
```

```
nextBtn.addEventListener("click", ()=> {
    allInput.forEach(input => {
        if(input.value != ""){
            form.classList.add('secActive');
        }else{
            form.classList.remove('secActive');
        }
    })
})

backBtn.addEventListener("click", () => form.classList.remove('secActive'));
</script>




<script>

    function load_car_models(company_id,car_model_id)
    {
        var company=document.getElementById(company_id);
        var car_model= document.getElementById(car_model_id);
        console.log(company.value);
        car_model.value="";
        car_model.innerHTML="";
        {% for company in companies %}
            if( company.value == "{{ company }}")
            {
                {% for model in car_models %}
                    {% if company in model %}

                        var newOption= document.createElement("option");
                        newOption.value="{{ model }}";
                        newOption.innerHTML="{{ model }}";
                        car_model.options.add(newOption);
                    {% endif %}
```

```
            {% endfor %}
        }
    {% endfor %}
}

function form_handler(event) {
    event.preventDefault(); // Don't submit the form normally
}
function send_data()
{
    document.querySelector('form').addEventListener("submit",form_handler);

    var fd=new FormData(document.querySelector('form'));

    var xhr= new XMLHttpRequest({mozSystem: true});

    xhr.open('POST','/predict',true);
    document.getElementById('prediction').innerHTML="Wait! Predicting Price.....";
    xhr.onreadystatechange = function(){
        if(xhr.readyState == XMLHttpRequest.DONE){
            document.getElementById('prediction').innerHTML="The Resale Value Predicted is:
₹"+xhr.responseText;

        }
    };

    xhr.onload= function(){};

    xhr.send(fd);
}
</script>


</body>
</html>
```

app.py

```
from flask import Flask, render_template, request, redirect
from flask_cors import CORS, cross_origin
import pickle
import pandas as pd
import numpy as np

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "8MK3A0eyRIQXgHpK0FjRMhfTXdaRMOhiGHVBmySZrQMh"
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}



app = Flask(name)
cors = CORS(app)

model = pickle.load(open('LinearRegressionModel.pkl', 'rb'))
car = pd.read_csv('Autos_Cleaned_Data.csv')



@app.route('/', methods=['GET', 'POST'])
def index():
    companies = sorted(car['company'].unique())
    car_models = sorted(car['name'].unique())
    vehicle_type = sorted(car['vehicle_type'].unique())
    year = sorted(car['year'].unique(), reverse=True)
    month = sorted(car['month'].unique())
    fuel_type = sorted(car['fuel_type'].unique())
    gear = sorted(car['gear'].unique())
    damaged = sorted(car['damaged'].unique())

    companies.insert(0, 'Select Company')
    vehicle_type.insert(0, 'Select Vehicle Type')
```

```
    fuel_type.insert(0, 'Select Fuel Type')
    year.insert(0, 'Select Year of Reg')
    month.insert(0, 'Select Month of Reg')
    gear.insert(0, 'Select the Gear Type')
    damaged.insert(0, 'Condition of the Car')


    return render_template('index.html', companies=companies, car_models=car_models, years=year,
                           vehicle_types=vehicle_type, months=month, fuel_types=fuel_type, gears=gear,
damages=damaged)


@app.route('/predict', methods=['POST'])
@cross_origin()
def predict():
    company = request.form.get('company')
    car_model = request.form.get('car_models')
    vehicle_type = request.form.get('vehicle_type')
    year = request.form.get('year')
    month = request.form.get('month')
    gear = request.form.get('gear')
    fuel_type = request.form.get('fuel_type')
    power = request.form.get('ps')
    kms_driven = int(request.form.get('kilo_driven'))
    damaged = request.form.get('damaged')
    print(company,car_model, vehicle_type, year, month, gear, fuel_type, power, kms_driven, damaged )


    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"fields": ['company', 'name', 'vehicle_type', 'year', 'month', 'gear',
'fuel_type', 'ps', 'kms_driven', 'damaged'], "values": [company,car_model, vehicle_type, year, month, gear,
fuel_type, power, kms_driven, damaged]}]}

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/0e796fd8-2ea1-
43f3-87a9-6b1ca049389f/predictions?version=2022-11-19', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
  predictions = response_scoring.json()
```

```
  print(response_scoring.json())
  prediction = predictions['predictions'][0]['values'][0][0]
  print(prediction)


    return str(np.round(prediction[0], 2))


if name == 'main':
    app.run()
```

GitHub Link:

GITHUB REPOSITORY LINK