**CAR RESALE VALUE PREDICTION**

**Team No:**PNT2022TMID34307

**Team Leader:** V.G.UMA LEKSHMI

**Team Members: V.R.VAISNAVI,S.NAVEENA KAVIN,B.SHAJINI**

**Project Report**

1. **INTRODUCTION** 
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
7. **CODING & SOLUTIONING**
8. **TESTING** 
   1. Test Cases
   2. User Acceptance Testing
9. **RESULTS**
   1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

GitHub & Project Demo Link

**1.INTRODUCTION**

* 1. **PROJECT OVERVIEW**

*Used car resale market in India was marked at 24.2 billion US dollars in 2019. Due to the huge requirement of used cars and lack of experts who can determine the correct valuation, there is an utmost need of bridging this gap between sellers and buyers. This project focuses on building a system that can accurately predict a resale value of the car based on minimal features like kms driven, year of purchase etc. without manual or human interference and hence it remains unbiased.*

In this project we have used different algorithms with different techniques for developing Car resale value prediction systems considering different features of the car. In a nutshell, car resale value prediction helps the user to predict the resale value of the car depending upon various features like kilometers driven, fuel type, etc.

Car resale value prediction system is made with the purpose of predicting the correct valuation of used cars that human intervention or manual examination, a user can predict the resale value of his car.

* 1. ***PURPOSE***

This resale value prediction system is made for general purpose to just predict the amount that can be roughly acquired by the user.

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.

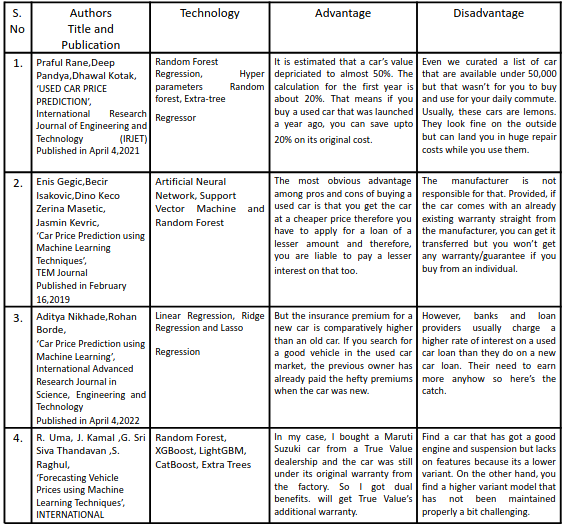
The main idea of making a car resale value prediction system is to get hands-on practice for python using Data Science. Car resale value prediction is the system to predict the amount of resale value based on the parameters provided by the user. User enters the details of the car into the form given and accordingly the car resale value is predicted.

**2.LITERATURE SURVEY**

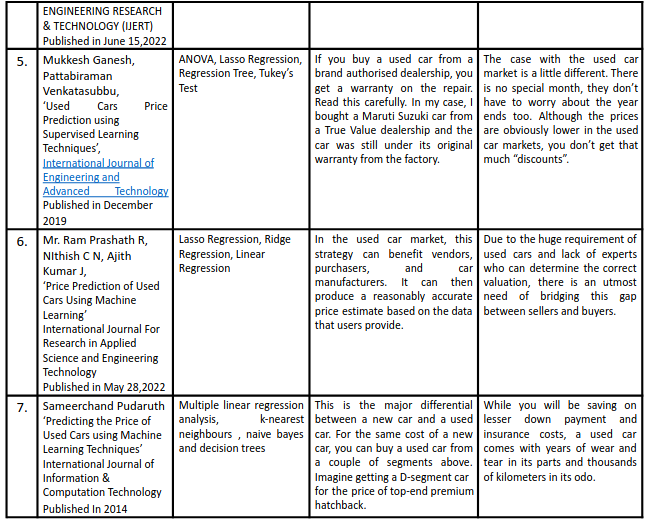
**2.1 EXISTING PROBLEM**

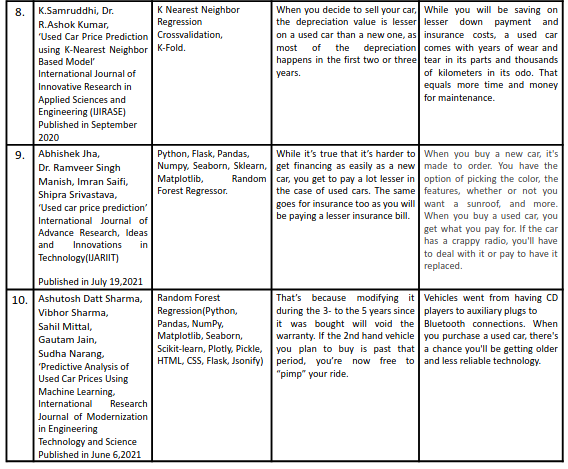
By author Sameer Chand, they have done the forecasts of vehicle cost from the chronicled information that has been gathered from every day papers. They have utilized the administered AI strategies for foreseeing the cost of vehicles. Numerous different calculations like various straight relapse, k-closest neighbor calculations, gullible based, and some choice tree calculations additionally been utilized. Every one of the four calculations are looked at and tracked down the best calculation for forecast. They have confronted a few challenges in looking at the calculations, by one way or another they have overseen. As indicated by creators Pattabiraman, this paper is more focused on the connection among vender and purchaser. To foresee the cost of four wheelers, more highlights are required like previously given value, mileage, make, model, trim, type, chamber, liter, entryways, voyage, sound, cowhide. Utilizing these highlights the cost of vehicle has been anticipated with the assistance of factual investigation framework for exploratory information examination. As per creators EnisGegic et al, in this paper the chiefly focus on gathering different information from web entryway by utilizing web scrap methods. Furthermore, those have been contrasted and the assistance of various AI calculations to foresee the vehicle cost in simple way. They arranged the value as per various scopes of value that is as of now given. Fake neural organization, support vector machine, arbitrary timberland calculations were utilized on various datasets to construct classifiers model. Another methodology was given by Richardson in his postulation work. In his hypothesis it states more strong vehicles will be delivered by vehicle maker. He looked at the crossover vehicles and conventional vehicles in scraper it really holds their incentive for longer time utilizing numerous relapse procedures. This works on the natural conditions, and furthermore it assists with giving colossal effectiveness of utilizing energizes. Wu et al, in this paper they have utilized neuro fluffy information based framework to exhibit vehicle value forecast. By considering the accompanying ascribes like brand, year of creation and sort of motor they anticipated a model which has comparative outcomes as the basic relapse model. Additionally, they made a specialist framework named ODAV (Optimal Distribution of Auction Vehicles) as there is a popularity for selling the by vehicles toward the finish of the renting year by vehicle vendors. This framework gives experiences into the best costs for vehicles, just as the area where all that cost can be acquired. To anticipate a cost of vehicles, the K – closest neighbor AI calculation has been utilized which depends on relapse models. More number of vehicles has been traded through this framework so this specific framework is all the more effectively oversaw.

**2.2 REFERENCES**









**2.3 PROBLEM STATEMENT DEFINITION**

The main aim of this project is to predict the price of used cars using the various Machine Learning (ML) models. This can enable the customers to make decisions based on different inputs or factors namely

• Brand or Type of the car one prefers like Ford, Hyundai

• Model of the car namely Ford Figo, Hyundai Creta

• Location like Delhi, Chennai, Mumbai

• Year of manufacturing like 2020, 2021

• Type of fuel namely Petrol, Diesel

• Price range or Budget

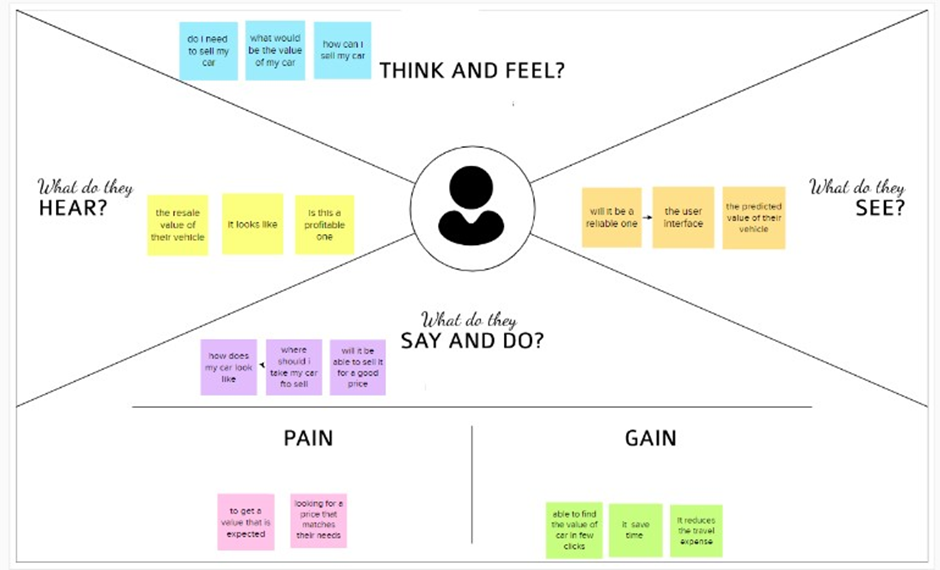
• Type of transmission which the customer prefers like Automatic or Manual

• Mileage

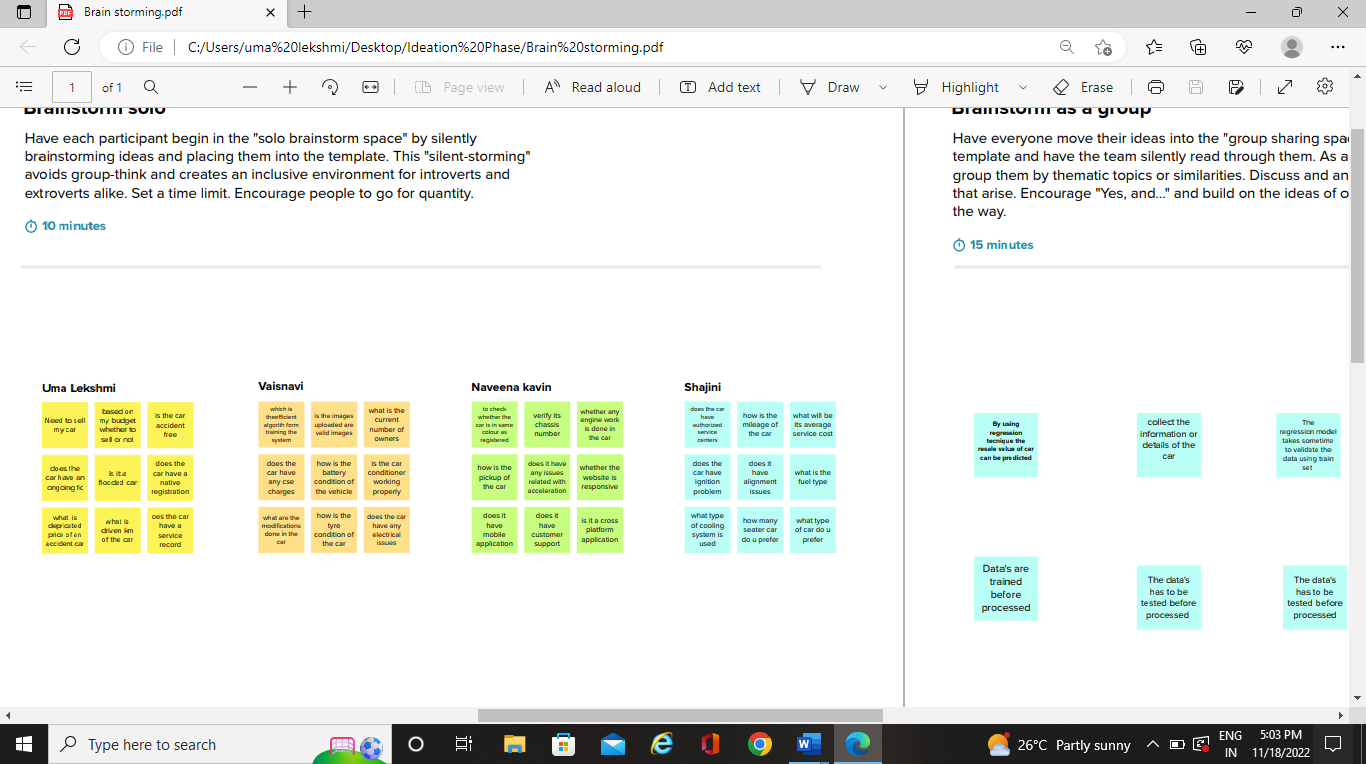
to name a few characteristic features required by the customer. The project Car Price Prediction deals with providing the solution to these problems. Through this project, we will get to know which of the factors are significant and tell us how they affect the car’s worth in the market.

**3.IDEATION & PROPOSED SOLUTION**

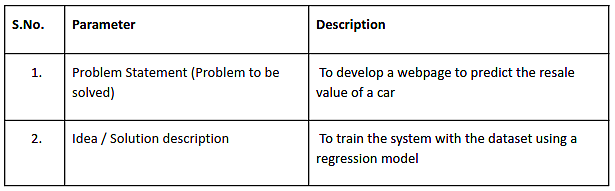
**3.1 EMPATHY MAP CANVAS**

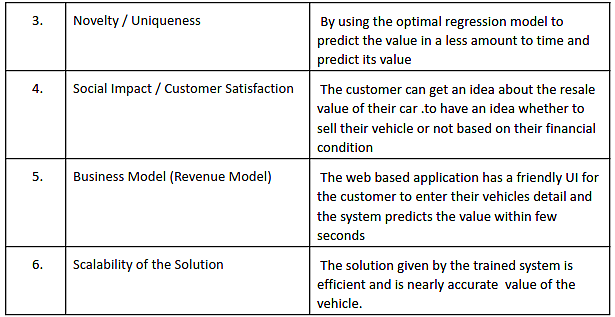


**3.2 IDEATION & BRAIN STORMING**

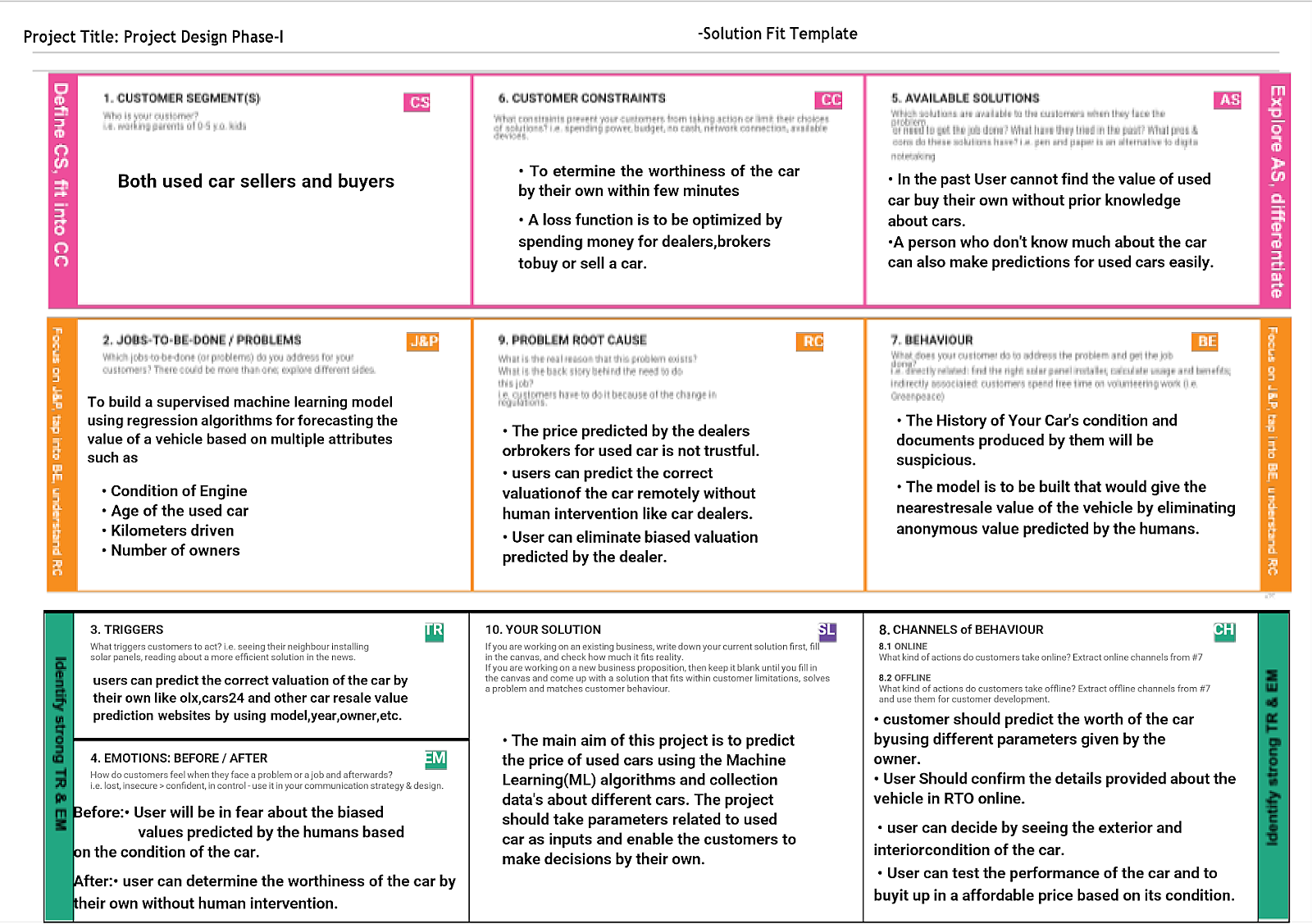


**3.3 PROPOSED SOLUTION**





**3.4 PROBLEM SOLUTION FIT**



**4.REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENTS**

**Navigator :**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

For this project, we will be using **Jupyter notebook and Spyder**

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

For this project, we will be using **Jupyter notebook and Spyder**

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

To build Machine learning models you must require the following packages

**Sklearn:** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.

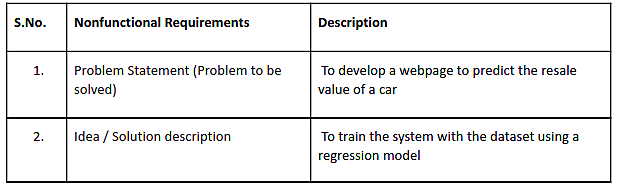
**NumPy**: NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object

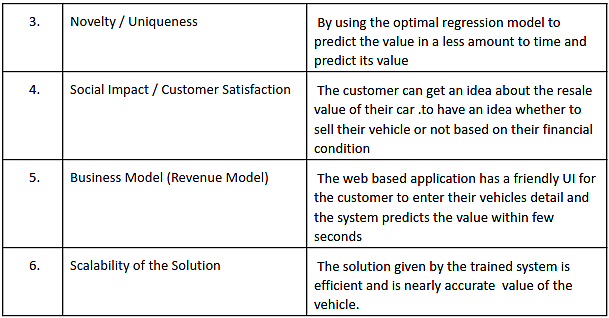
**Pandas**: pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

**Matplotlib**: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

**Flask**: Web framework used for building Web applications.

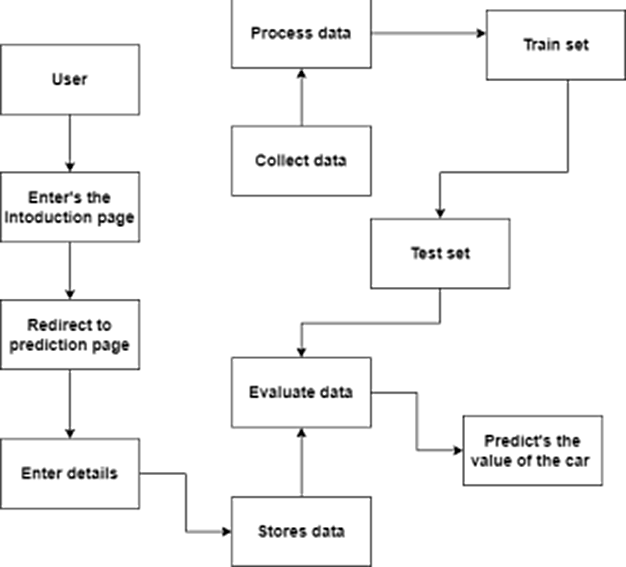
**4.2 NONFUNCTIONAL REQUIREMENTS**



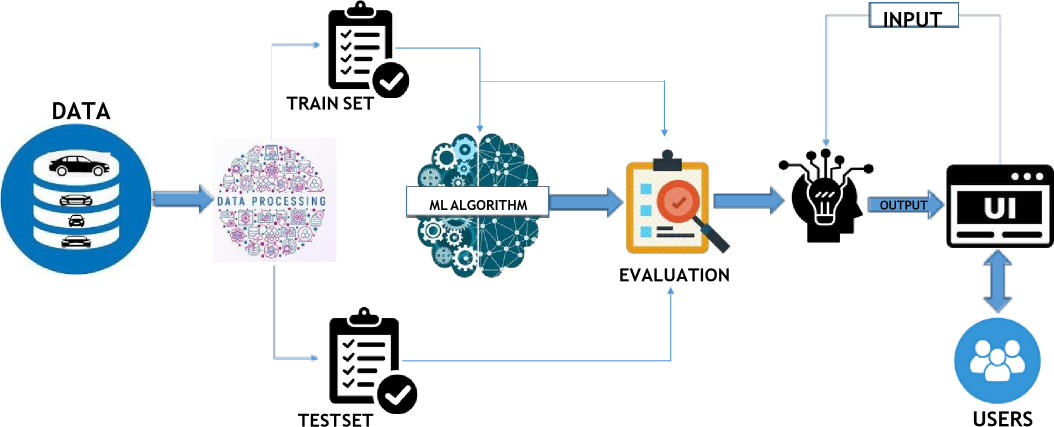


**5.PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAM**



**5.2 SOLUTION TECHICAL ARCHITECTURE**

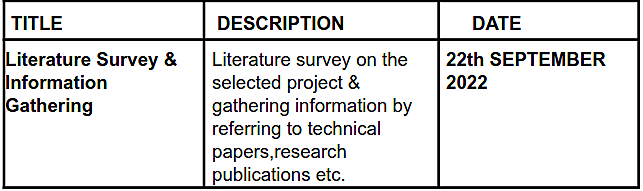


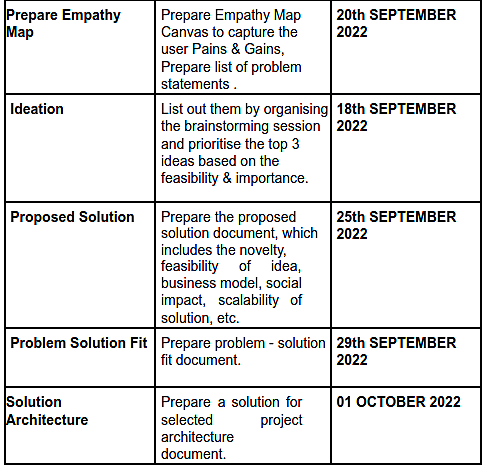
**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional**  **Requirement**  **(Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (web user) | Enters the browser | USN-1 | As a user, I can access to website using a web browser | I can enter by selecting the  appropriate web link | 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 acceptancce | 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 |  |  |  |  |  |  |

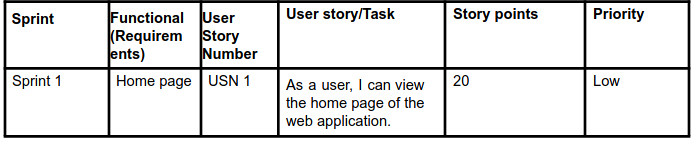
**6.PROJECT PLANNING & SCHEDULING**

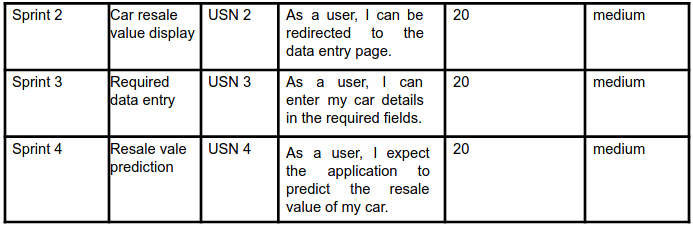
**6.1 SPRINT PLANNING & ESTIMATION**





**6.2 SPRINT DELIVERY SCHEDULE**





**7.CODING & SOLUTIONING**

#import libraries import pandas as pd import numpy as np import matplotlib as plt from sklearn.preprocessing import LabelEncoder import pickle # read the dataset

df = pd.read\_csv("Data/autos.csv", header=0, sep=',', encoding='Latin1',) df # clean the dataset

df=df.drop('offerType',axis=1)

df=df[(df.powerPS > 50) & (df.powerPS < 900)] df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)] df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen',

'postalCode','dateCreated'], axis='columns',inplace=True) new\_df = df.copy() new\_df = new\_df.drop\_duplicates ([ 'price', 'vehicleType',

'yearOfRegistration'

,'gearbox', 'powerPS', 'model',

'kilometer', 'monthOfRegistration', 'fuelType'

,'notRepairedDamage'])

new\_df.gearbox.replace(('manuell', 'automatik'), ('manual', 'automatic'), inplace=True) new\_df.fuelType.replace(('benzin', 'andere', 'elektro'), ('petrol',

'others', 'electric'), inplace=True)

new\_df.vehicleType.replace(('kleinwagen', 'cabrio', 'komb, 'andere'),

('small car', 'convertible', 'combination',

'others'), inplace=True)

new\_df.notRepairedDamage.replace(('ja', 'nein'), ('Yes',

'No'),inplace=True)

new\_df = new\_df[(new\_df.price >= 100) & (new\_df.price <= 150000)] new\_df['notRepairedDamage'].fillna(value='not-declared', inplace=True) new\_df[ 'fuelType'].fillna(value='not-declared', inplace=True) new\_df[ 'gearbox'].fillna(value='not-declared', inplace=True) new\_df[ 'vehicleType'].fillna (value='not-declared', inplace=True) new\_df['model'].fillna(value='not-declared',inplace=True) new\_df.to\_csv("autos\_preprocessed.csv")

#import libraries import pandas as pd import numpy as np from sklearn.preprocessing import LabelEncoder from sklearn.model\_selection import train\_test\_split from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score import pickle

from lightgbm import LGBMRegressor #read preprocessed data data = pd.read\_csv("autos\_preprocessed.csv") #metrics evaluation

def find\_scores(Y\_actual, Y\_pred, X\_train):

scores = dict()

mae = mean\_absolute\_error(Y\_actual, Y\_pred)

mse = mean\_squared\_error(Y\_actual, Y\_pred)

rmse = np.sqrt(mse)

rmsle = np.log(rmse)

r2 = r2\_score(Y\_actual, Y\_pred)

n, k = X\_train.shape

adj\_r2\_score = 1 - ((1-r2)\*(n-1)/(n-k-1))

scores['mae']=mae

scores['mse']=mse

scores['rmse']=rmse

scores['rmsle']=rmsle

scores['r2']=r2

scores['adj\_r2\_score']=adj\_r2\_score

return scores

#testing abd training

1. = labeled.iloc[:,1:].values
2. = labeled.iloc[:,0].values.reshape(-1,1)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.4, random\_state=42)

model =

LGBMRegressor(boosting\_type="gbdt",learning\_rate=0.07,metric="rmse",n\_esti mators=300,objective="root\_mean\_squared\_error",random\_state=42,reg\_sqrt=Tr ue) model.fit(X\_train, Y\_train) Y\_pred = model.predict(X\_test) find\_scores(Y\_test, Y\_pred, X\_train)

#save the model

pickle.dump(model, open('resale\_model.sav', 'wb'))

BUILD A FLASK APP

# Import Libraries import pandas as pd import numpy as np

from flask import Flask, render\_template, Response, request import pickle

from sklearn.preprocessing import LabelEncoder app = Flask(\_\_name\_\_)#initiate flask app

def load\_model(file='resale\_model.sav'):#load the saved model return pickle.load(open(file, 'rb'))

@app.route('/') def index():#main page return render\_template('car.html')

@app.route('/predict\_page') def predict\_page():#predicting page return render\_template('value.html')

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

reg\_year = int(request.args.get('regyear')) powerps = float(request.args.get('powerps')) kms= float(request.args.get('kms')) reg\_month = int(request.args.get('regmonth')) gearbox = request.args.get('geartype') damage = request.args.get('damage') model = request.args.get('model') brand = request.args.get('brand') fuel\_type = request.args.get('fuelType') veh\_type = request.args.get('vehicletype')

new\_row = {'yearOfReg':reg\_year, 'powerPS':powerps, 'kilometer':kms,

'monthOfRegistration':reg\_month, 'gearbox':gearbox,

'notRepairedDamage':damage,

'model':model, 'brand':brand, 'fuelType':fuel\_type,

'vehicletype':veh\_type}

print(new\_row) new\_df = pd.DataFrame(columns=['vehicletype','yearOfReg','gearbox',

'powerPS','model','kilometer','monthOfRegistration','fuelType',

'brand','notRepairedDamage']) new\_df = new\_df.append(new\_row, ignore\_index=True) labels =

['gearbox','notRepairedDamage','model','brand','fuelType','vehicletype'] mapper = {}

for i in labels:

mapper[i] = LabelEncoder()

mapper[i].classes = np.load(str('classes'+i+'.npy'),

allow\_pickle=True) transform = mapper[i].fit\_transform(new\_df[i]) new\_df.loc[:,i+'\_labels'] = pd.Series(transform,

index=new\_df.index) labeled =

new\_df[['yearOfReg','powerPS','kilometer','monthOfRegistration'] +

[x+'\_labels' for x in labels]]

X = labeled.values.tolist() print('\n\n', X) predict = reg\_model.predict(X)

#predict =predictions['predictions'][0]['values'][0][0] print("Final prediction :",predict)

return render\_template('predict.html',predict=predict)

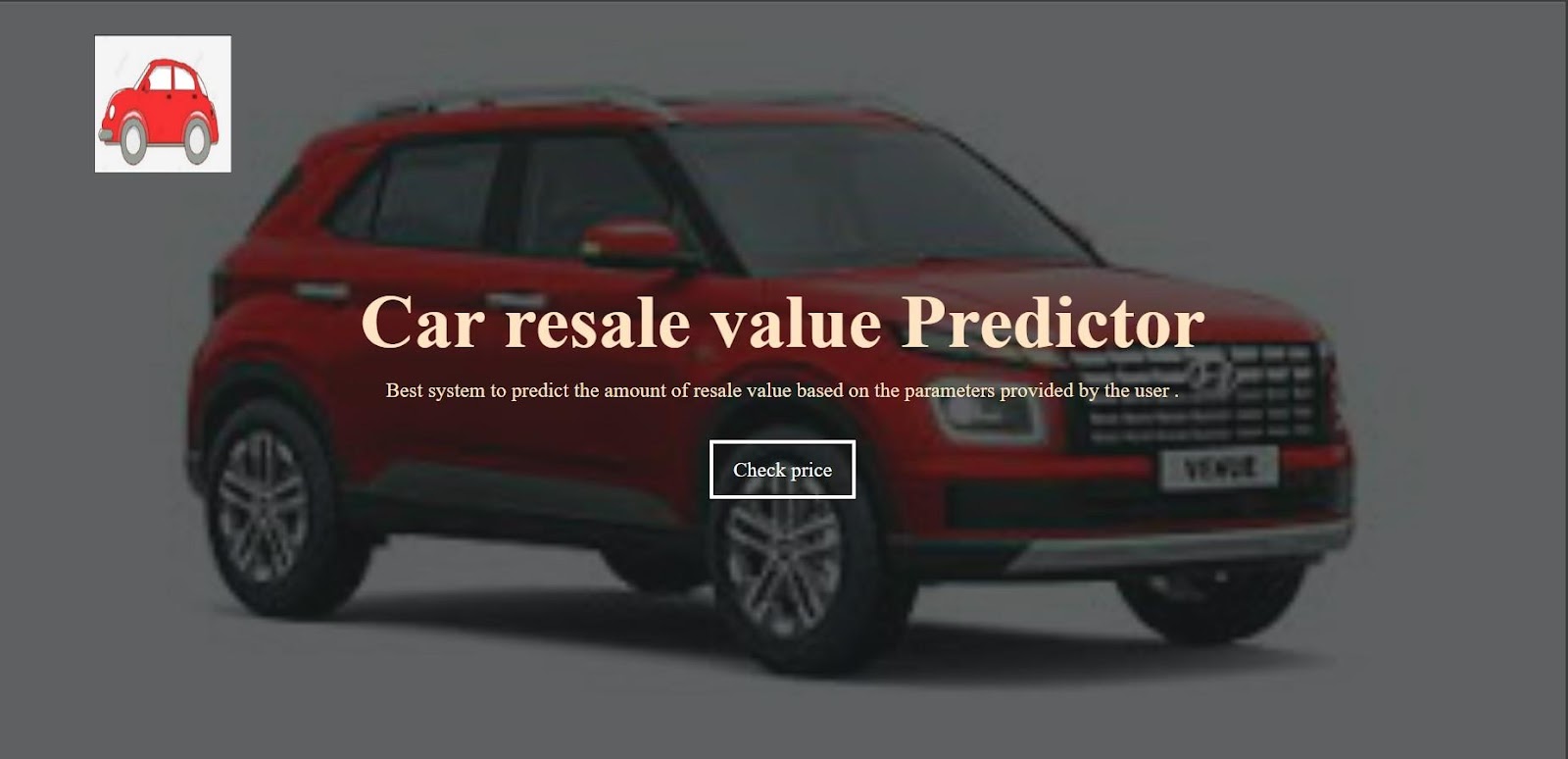
if \_\_name\_\_=='\_\_main\_\_':

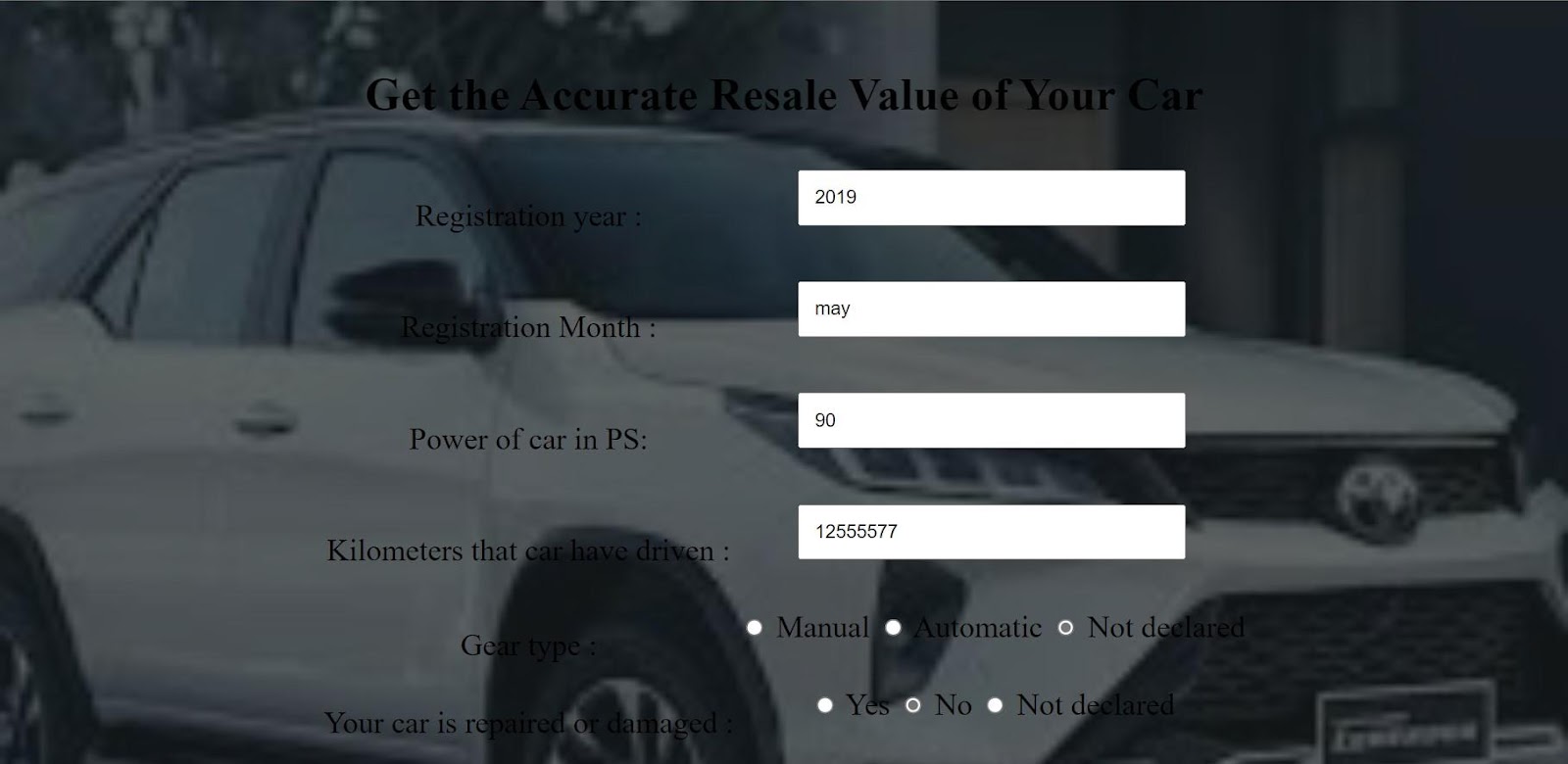
reg\_model = load\_model()#load the saved model app.run(debug=True)

**8.TESTING**

**8.1.TEST CASES**

1. User Login and Registration test
2. Database Update test
3. Prediction test

**8.2.USER ACCEPTANCE TESTING** 





**10.ADVANTAGES & DISADVANTAGES**

**MERITS:**

Highly effective

Time efficient

Less power consumption

More information obtained

**DEMERITS:**

Not accurate

Not effective

**11.CONCLUSION**

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction. This project compares 2 different algorithms for machine learning : Decision tree, Random forest.

**12.FUTURE SCOPE**

In future this machine learning model may bind with various website which can provide real time data for price prediction. Also we may add large historical data of car price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

**12.APPENDIX**

1. **app.py**

# Import Libraries import pandas as pd import numpy as np

from flask import Flask, render\_template, Response, request import pickle from sklearn.preprocessing import LabelEncoder

app = Flask(\_\_name\_\_)#initiate flask app

def load\_model(file='resale\_model.sav'):#load the saved model return pickle.load(open(file, 'rb'))

@app.route('/') def index():#main page return render\_template('car.html')

@app.route('/predict\_page') def predict\_page():#predicting page return render\_template('value.html')

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

reg\_year = int(request.args.get('regyear')) powerps = float(request.args.get('powerps')) kms= float(request.args.get('kms')) reg\_month = int(request.args.get('regmonth'))

gearbox = request.args.get('geartype') damage = request.args.get('damage') model = request.args.get('model') brand = request.args.get('brand') fuel\_type = request.args.get('fuelType') veh\_type = request.args.get('vehicletype')

new\_row = {'yearOfReg':reg\_year, 'powerPS':powerps, 'kilometer':kms,

'monthOfRegistration':reg\_month, 'gearbox':gearbox,

'notRepairedDamage':damage,

'model':model, 'brand':brand, 'fuelType':fuel\_type,

'vehicletype':veh\_type}

print(new\_row) new\_df = pd.DataFrame(columns=['vehicletype','yearOfReg','gearbox',

'powerPS','model','kilometer','monthOfRegistration','fuelType',

'brand','notRepairedDamage']) new\_df = new\_df.append(new\_row, ignore\_index=True) labels =

['gearbox','notRepairedDamage','model','brand','fuelType','vehicletype'] mapper = {}

for i in labels:

mapper[i] = LabelEncoder()

mapper[i].classes = np.load(str('classes'+i+'.npy'),

allow\_pickle=True) transform = mapper[i].fit\_transform(new\_df[i]) new\_df.loc[:,i+'\_labels'] = pd.Series(transform,

index=new\_df.index) labeled =

new\_df[['yearOfReg','powerPS','kilometer','monthOfRegistration'] +

[x+'\_labels' for x in labels]]

X = labeled.values.tolist() print('\n\n', X) predict = reg\_model.predict(X)

#predict = predictions['predictions'][0]['values'][0][0] print("Final prediction :",predict) return render\_template('predict.html',predict=predict)if \_\_name\_\_=='\_\_main\_\_':

reg\_model = load\_model()#load the saved model app.run(debug=True)

**GITHUB :**

https://github.com/IBM-EPBL/IBM-Project-50672-1660921137

**PROJECT DEMO LINK:** <https://drive.google.com/drive/folders/1GGPeBLsK7LSpdoAF5KWaFrNkMziAc7OS>