PROJECT REPORT

1. INTRODUCTION

a. Project Overview

To be able to predict used cars market value can help both buyers and sellers. There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it’s a big corner to pay too much or sell less then it’s market value. In this Project, we are going to predict the Price of Used Cars using various features like year, model type, brand, fuel type, kilo-meter. Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and its value in the present-day scenario. In fact, seller also has no idea about the car’s existing value or the price he should be selling the car at. To overcome this problem, we have developed a model which will be highly effective. Gradient boosting Regressor is used because calculates the difference between the current prediction and the known correct target value. Because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user’s inputs.

b. Purpose

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. Car resale value prediction system is made with the purpose of predicting the correct valuation of used cars that helps users to sell the car remotely with perfect valuation and without human intervention in the process to eliminate biased valuation.

2. LITERATURE SURVEY

a. Existing problem

Predicting the price of used cars using machine learning Techniques has worked on analysing various supervised learning algorithms for predicting the resale value of the cars used in Mauritius. Comparative study on KNN, regression, naïve bayes and decision tree has been made to come up with high accuracy. Prediction of Resale Value of the Car Using Linear Regression Algorithm has used linear regression algorithm to estimate the car resale value. This research work has been implemented for accurately predicting the resale value of the vehicle based on the most significant attributes which are selected based on the highest correlation. This gives 90% percent accuracy and error obtained is 10%. Used car price prediction using K- Nearest Neighbour Based Model has used K nearest Neighbour algorithm to predict the resale value of the used cars. It has fetched around 85% accuracy. This model has also validated with 5 and 10 folds by using K Fold Method.

b. Reference

1. “Gegic, Enis, et al. "Car price prediction using machine learning techniques." *TEM Journal* 8.1 (2019): 113.

2. “Das Adhikary, Dibya Ranjan, Ronit Sahu, and Sthita Pragyna Panda. "Prediction of Used Car Prices Using Machine Learning." *Biologically Inspired Techniques in Many Criteria Decision Making*. Springer, Singapore, 2022. 131-140.

3. “Samruddhi, K., and R. Ashok Kumar. "Used Car Price Prediction using K-Nearest Neighbor Based Model." *Int. J. Innov. Res. Appl. Sci. Eng.(IJIRASE)* 4 (2020): 629-632.

4. “Gajera, Prashant, Akshay Gondaliya, and Jenish Kavathiya. "Old Car Price Prediction With Machine Learning." *Int. Res. J. Mod. Eng. Technol. Sci* 3 (2021): 284-290.

5. “C. V. Narayana, N. O. G. Madhuri, A. NagaSindhu, M. Aksha and C. Naveen, "Second Sale Car Price Prediction using Machine Learning Algorithm," ***2022 7th International Conference on Communication and Electronics Systems (ICCES)***, 2022, pp. 1171-1177, doi:

10.1109/ICCES54183.2022.9835872.

c. Problem Statement Definition

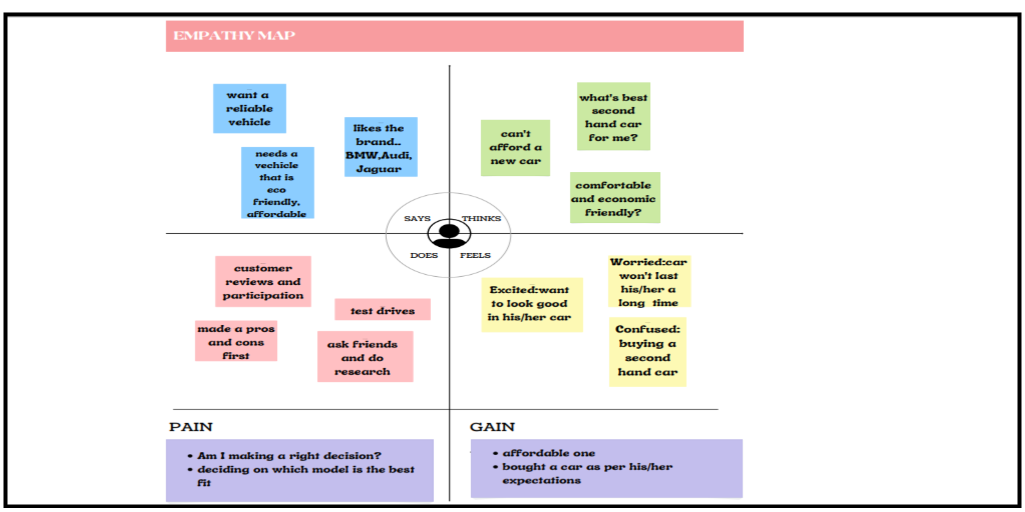
It is easy for any company to price their new cars based on the manufacturing and marketing cost it involves. But when it comes to a used car it is quite difficult to define a price because it involves it is influenced by various parameters like car brand, manufactured year and etc. The goal of our project is to predict the best price for a pre-owned car in the Indian market based on the previous data related to sold cars using machine learning.

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

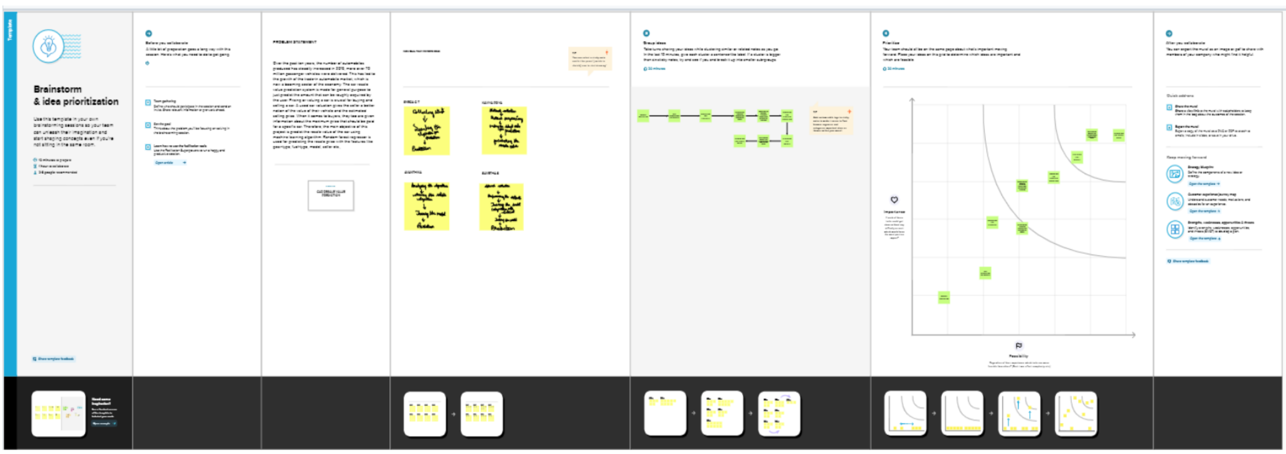
Empathy maps are a straightforward, effective technique for developing knowledge of your people. Empathy, the capacity to comprehend another person's feelings and thoughts, is the name's etymological source. When grounded in actual data and used in conjunction with other mapping techniques, they can:

- Eliminate bias from our designs and bring the team together around a single, shared knowledge of the user
- Find the gaps in our study's findings
- Find out what the user needs—needs that the user may not even be aware of
- Learn what motivates user action. Point us in the direction of genuine innovation



b. Ideation & Brainstorming

By posing a problem to a group of individuals or team members and engaging them in an open dialogue, the brainstorming approach allows for the generation of ideas. Agile Brainstorming is the name given to this method when it is used in agile projects since it may provide creative ideas. Our group speaks aloud each danger as it is identified. They can take notes so they won't forget a concept before their turn if an increased risk prompts a fresh thought for someone who is not yet in line.

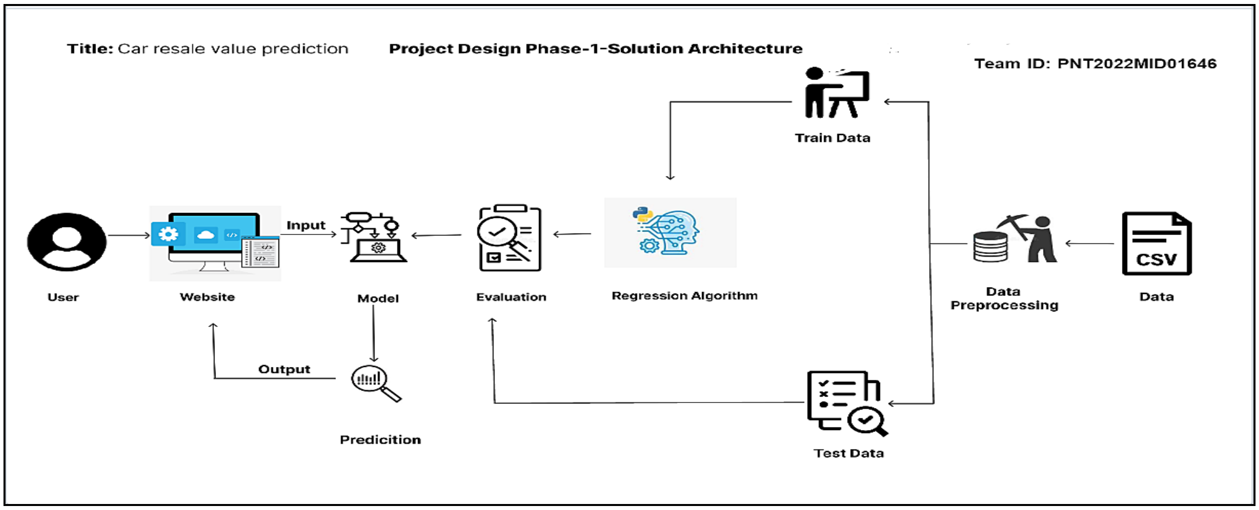


c. Proposed Solution

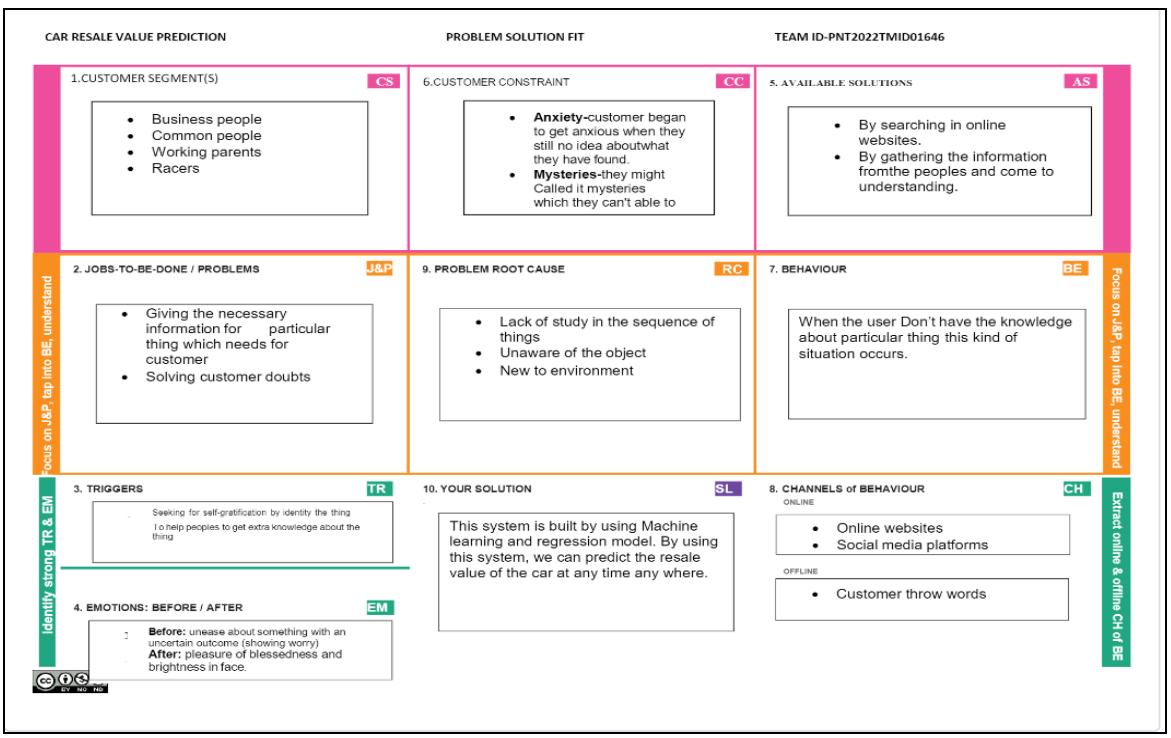
S.No.	Parameter	Description

1.	Problem Statement (Problem to besolved)	1. Sales prediction is the current numeroustrend in which all the business companiesthrive and it also aids the organization or concern in determining the future goals for it and its plan and procedure to achieve it. 2. Resales of cars almostoccupy a majorpart in everysales economy. 3. In that regard various factors like registration year, engine condition, company service record, spare parts condition, tyre condition, car body condition, kilometers covered, Interior look, color, mileage, number of owners, battery condition are taken into consideration before buying it along with engine condition and insurance. 4. The predication using thefactors would suggest the final productto be brought. 5. But these data may be inaccurate at times and there is a need of a proper algorithm that will provide a result withgood accuracy rate.
2.	Idea / Solution description	1. The overall proposed idea is to predict the car resale value and show it to the required people. 2. This idea can be implemented and could be presented to the customer. This involves two phases. 3. One phase is collecting the dataset for training the car resale value prediction model. 4. Testing the car resale value prediction model. 5. The second phase involves creating a website (front end) for presenting the entire solution as a customized GUI so that this would be very useful for the user to utilize this solution

		6. The user will be asked to enter the details for prediction like model, price, design, kilometres covered, Interior look, colour. 7. If user clicks the predict option, the predicted resale value will be displayed in the website.
3.	Novelty / Uniqueness	1. Consumer behavior changes, it's a fact. So for betteraccuracy select amore recently added product whenpossible. 2. You can use multiple reference products to getthe best averageand the noveltysales estimates willbe based on features from all of them using the average.
4.	Social Impact / Customer Satisfaction	1. Sales forecasting helps you attainthis revenue efficiency by offeringinsight into the likely behavior of your mostvaluable customers. 2. You can predict futuresales, as well as improve pricing, advertising,and product development.
5.	Scalability of the Solution	1. Here we are using time series analysis so, When historical data for a product or product line is available and patterns areobvious, organisations typically employ the time series analysis technique to demand forecasting. 2. A time seriesanalysis can help youdetect seasonal variations in demand, cyclical patterns, and majorsales trends. 3. The time series analysis approach works best for well-established organisations with several years ofdata to work with and very steady trendpatterns.



d.Problem Solution Fit



4. REQUIREMENTS ANALYSIS:

a.Functional requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Application	Filling of application Modification of application Verification of application
FR-4	Loan Issuance	Checking status of loan Loan Approval Loan Rejection
FR-5	Credit history analysis	Credit score auditing Income auditing
FR-6	User management	Choosing appropriate loan program for users Categorising users according to credit history.

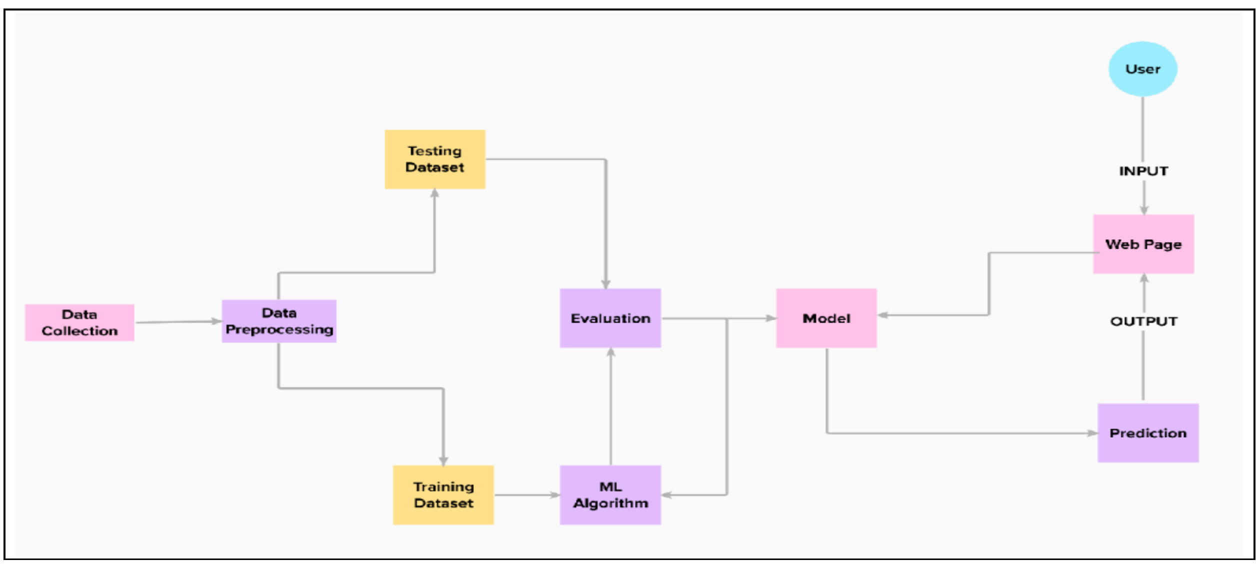
b. Non - functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Simple and understandable UI. Easy to navigate Smooth and seamless

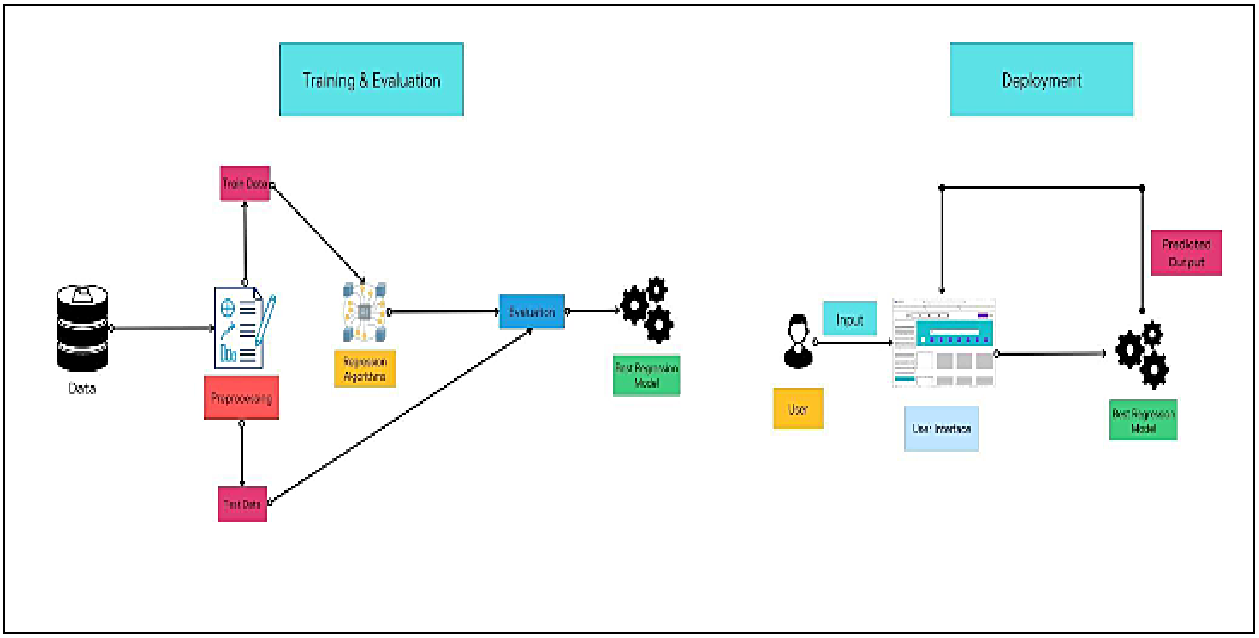
		Easy to comprehend
NFR-2	**Security**	Restricted access to data. Login verification Registration verification Upholding privacy of user
NFR-3	**Reliability**	Backup to prevent data loss Negation of data loss due to lag.
NFR-4	**Performance**	Web based application. Requires minimum Intel Pentium 4 processor, 4 GB RAM, 1280x1024 screen with application window size 1024x680

5. PROJECT DESIGN

a. Data flow diagrams



b. Solution & Technical Architecture



c. User stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Data Entry	USN-1	As a user, I can enter the car details in the application.	I can enter the car details	Medium	Sprint-1
Customer (Mobile user)	Obtain output	USN-2	As a user, I will receive car resale value in the application.	I can receive my carresale value	High	Sprint-1
Customer (Mobile user)	Data Entry	USN-1	As a user, I can enter the car details in the application.	I can enter the car details	Medium	Sprint-1
Customer (Mobile user)	Obtain output	USN-2	As a user, I will receive car resale value in the application.	I can receive my carresale value	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

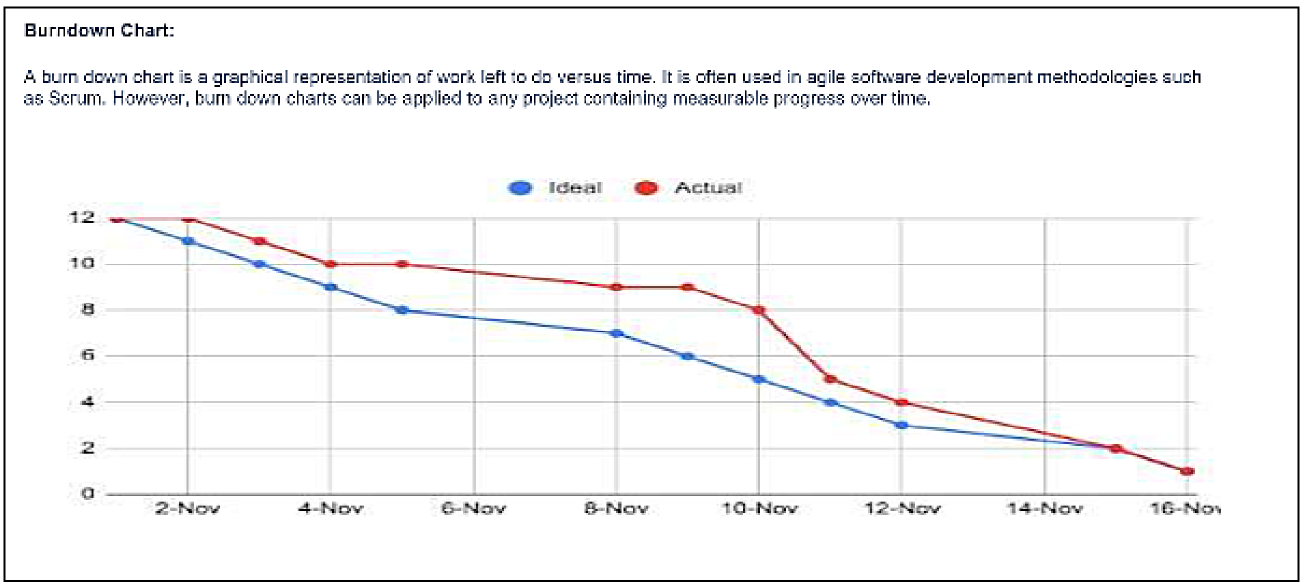
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Pre- process the data	USN-1	Collect and download the Dataset	2	High	Avanthigaa P
Sprint-1	Pre- process the data	USN-2	Import required libraries	1	High	Suvetha S
Sprint-1	Pre- process the data	USN-3	Read and cleanthe dataset	2	Low	Snega G T
Sprint-2	Model Building	USN-1	Split the data into independent and dependent variables	2	Medium	Kaviyapriya P
Sprint-2	Model Building	USN-2	Build regression model	1	High	Kaviyapriya &Snega G T
Sprint-3	Application Building	USN-1	Build pythonapplication	2	Medium	Suvetha S & Avanthigaa P
Sprint-3	Application Building	USN-2	Test theapplication model	3	High	Snega G T & Suvetha S
Sprint-4	Train the model	USN-1	Train the model	3	High	Avanthigaa P

b. Sprint Delivery Schedule

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 Nov2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

c. Reports from JIRA



7. CODING & SOLUTIONING

a. Feature 1 - Collect the dataset and preprocess the data.

Machine Learning has become a tool used in almost every task that requires estimation. So we need to build a model to estimate the price of used cars. The model should take car-related parameters and output a selling price. On sprint-1 the selling price of a used car depends on certain features datasets are collected from different open sources like kaggle.com, data.gov, UCI machine learning repository, the dataset which contains a set of features through which the resale price of the car can be identified is to be collected as

- price
- vehicle Type
- year Of Registration
- gearbox
- model
- kilo meter
- month Of Registration
- fuel Type
- brand
- not Repaired Damage

ML is a data hunger technology, it depends heavily on data, without data, it is impossible. It is the most crucial aspect that makes algorithm training possible. Collects Data, Import necessary packages, Pre-process images, and passes on to Network Model and Saves Model Weights. The libraries can be imported,

Pre-Process The Data:

Pre-processing the dataset that includes:

1. Handling the null values.

2. Handling the categorical values if any.

3. Normalize the data if required.

4. Identify the dependent and independent variables.

Data cleaning and wrangling methods are applied on the *used cars* data file. Before making data cleaning, some explorations and data visualizations were applied on data set. This gave some idea and guide about how to deal with missing values and extreme values. After data cleaning, data exploration was applied again in order to understand cleaned version of the data.

```
import pandas as pd
import numpy as np
import matplotlib as plt
from sklearn.preprocessing import LabelEncoder
import pickle
```

```
df=pd.read_csv("autos.csv",header=0,sep=',',encoding='latin',)
```

```
In [22]: df.columns

Out[22]: Index(['dateCrawled', 'name', 'offerType', 'price', 'abtest', 'vehicleType',
                'yearOfRegistration', 'gearbox', 'powerPS', 'model', 'kilometer',
                'monthOfRegistration', 'fuelType', 'brand', 'notRepairedDamage',
                'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen'],
               dtype='object')
```

```
In [24]: print(df.seller.value_counts())
         df[df.seller != 'gewerblich']
         df=df.drop('seller',1)
         print(df.offerType.value_counts())
         df[df.offerType != 'Gesuch']
         df=df.drop ('offerType',1)

         privat        371525
         gewerblich         3
         Name: seller, dtype: int64
         Angebot    371516
         Gesuch         12
         Name: offerType, dtype: int64

         /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: In a future version of pandas all arguments of D
         ataFrame.drop except for the argument 'labels' will be keyword-only
           This is separate from the ipykernel package so we can avoid doing imports until
         /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: FutureWarning: In a future version of pandas all arguments of D
         ataFrame.drop except for the argument 'labels' will be keyword-only
```

```
print(df.shape)
df = df[(df.powerPS > 50) & (df.powerPS <900)]
print(df.shape)
print(df.shape)
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)
```

```
(371528, 18)
(319709, 18)
(319709, 18)
(309171, 18)
```

```
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen','postalCode', 'dateCreated'], axis='columns', inplace=True)
```

```
new_df = df.copy()
```

```
df.columns
```

```
Index(['price', 'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS',
       'model', 'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage'],
      dtype='object')
```

```
new_df.columns
```

```
Index(['price', 'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS',
       'model', 'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage'],
      dtype='object')
```

```
new_df = new_df.drop_duplicates(['price', 'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS', 'model', 'kilometer', 'mont
```

```
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', 'kombi', 'andere'),('small car', 'convertible', 'combination', 'others'), in
new_df.notRepairedDamage.replace(('ja', 'nein'), ('Yes', 'No'), inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  return self._update_inplace(result)
```

```
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)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:6392: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  return self._update_inplace(result)
```

```
new_df.to_csv("autos_preprocessed.csv")
```

```
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
mapper = {}
for i in labels:
  mapper[i] = LabelEncoder()
  mapper[i].fit(new_df[i])
  tr = mapper[i].transform(new_df[i])
  np.save(str('classes'+i+'.npy'), mapper[i].classes_)
  print(i,":", mapper[i])
  new_df.loc[:, i + '_labels'] = pd. Series (tr, index=new_df.index)

labeled = new_df[ ['price','yearOfRegistration' , 'powerPS' ,'kilometer' , 'monthOfRegistration']+ [x+"_labels" for x in labels]]
print(labeled.columns)
```

```
gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
Index(['price', 'yearOfRegistration', 'powerPS', 'kilometer',
       'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
       'model_labels', 'brand_labels', 'fuelType_labels',
       'vehicleType_labels'],
      dtype='object')
```

b. Feature 2 - Training and testing and creating the application

A training model is a dataset that is used to train an algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training is the process of feeding an algorithm with data to help identify and learn good values for all

```python
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

#regression model
from lightgbm import LGBMRegressor
```

Import Preprocessed Data

```python
data = pd.read_csv('autos_preprocessed.csv')
data.head()
```

	Unnamed: 0	price	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage
0	1	18300	coupe	2011	manual	190	not-declared	125000	5	diesel	audi	Yes
1	2	9800	suv	2004	automatic	163	grand	125000	8	diesel	jeep	not-declared
2	3	1500	small car	2001	manual	75	golf	150000	6	petrol	volkswagen	No
3	4	3600	small car	2008	manual	69	fabia	90000	7	diesel	skoda	No
4	5	650	limousine	1995	manual	102	3er	150000	10	petrol	bmw	Yes

```python
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']

mapper = {}
for i in labels:
    mapper[i] = LabelEncoder()
    mapper[i].fit(data[i])
    tr = mapper[i].transform(data[i])
    np.save(str('classes'+i+'.npy'), mapper[i].classes_)
    data.loc[:, i+'_labels'] = pd.Series(tr, index=data.index)

labeled = data[['price', 'yearOfRegistration','powerPS','kilometer','monthOfRegistration']
                    +[x+"_labels" for x in labels]]

print(labeled.columns)
```

```
Index(['price', 'yearOfRegistration', 'powerPS', 'kilometer',
       'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
       'model_labels', 'brand_labels', 'fuelType_labels',
       'vehicleType_labels'],
      dtype='object')
```

Different Metrics Evaluation

```python
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
```

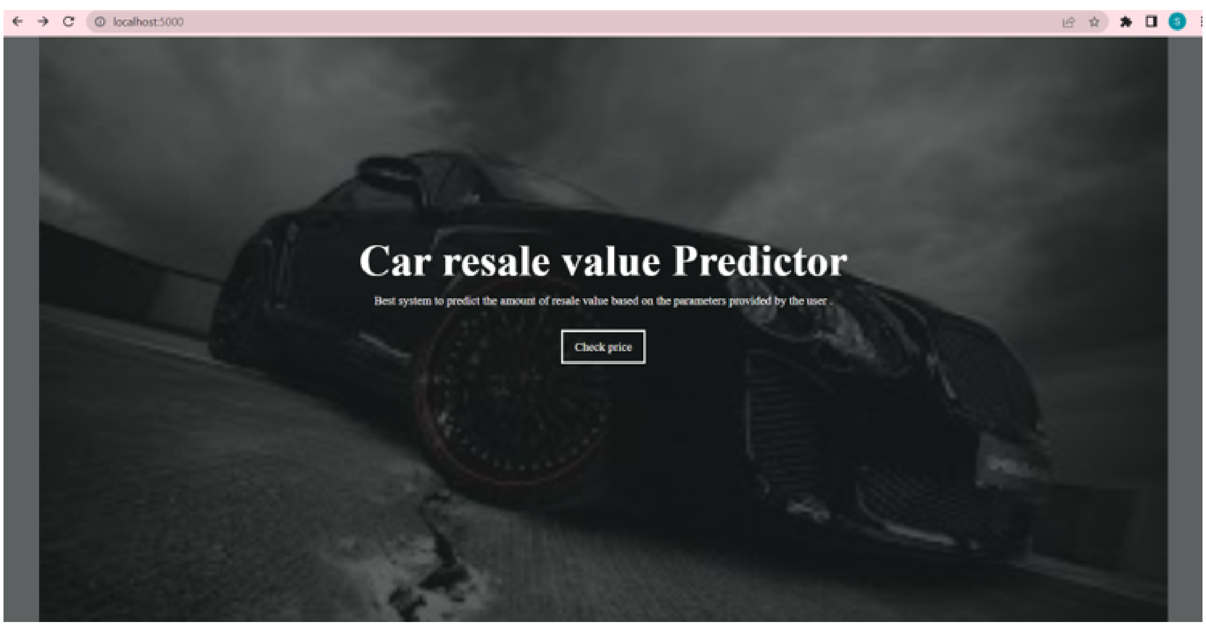
Train Test Split

```
X = labeled.iloc[:,1:].values
Y = labeled.iloc[:,0].values.reshape(-1,1)
```

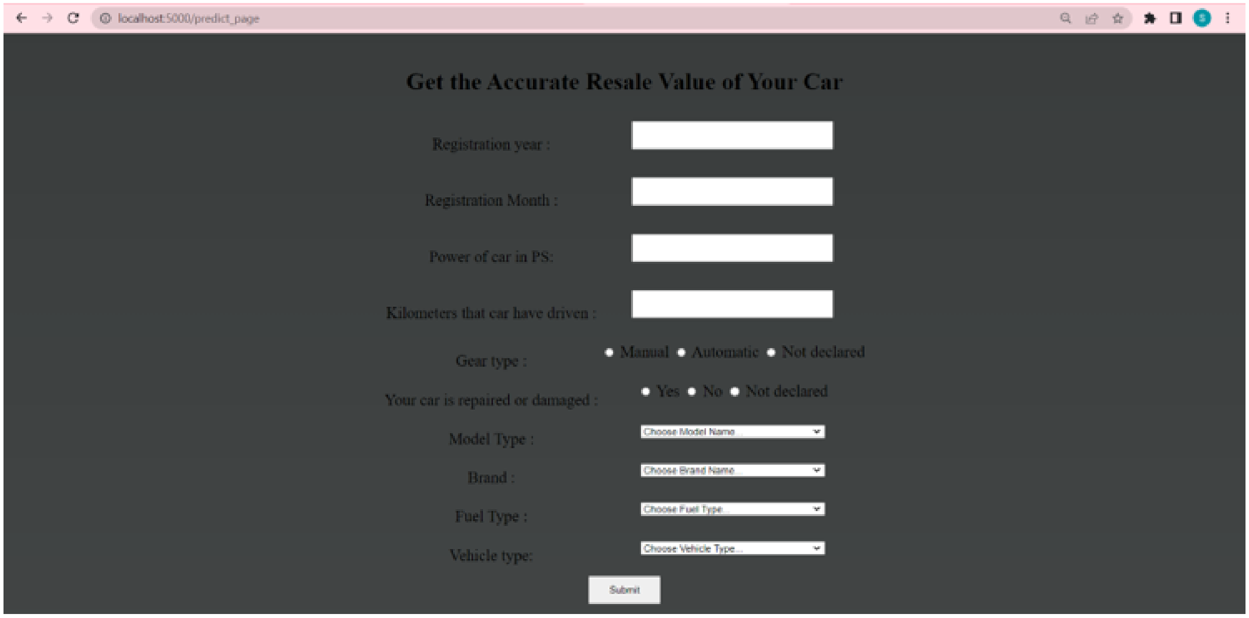
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4, random_state=42)
```

8. RESULTS:

Home page



Prediction form page:



9. ADVANTAGES AND DISADVANTAGES:

Advantages:

The car resale value prediction system helps in predicting the resale value of the car in a much easier manner. This helps the customers to easily predict the value of the car for resaling it.

Disadvantages:

This system will not be known to most of the users who are not prominent in using the online platforms for car resaling. Therefore, the possibility of using this system by many users is not possible.

10. CONCLUSION:

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. By performing ML models, we aim to get a better result or less error with max accuracy to predict the value of the used car. Initially, data cleaning is performed to remove the null values and outliers from the dataset then ML models are implemented to predict the price of cars. Next, with the help of data visualization features were explored deeply. The relation between the features is examined. From the report, it can be said that gradient regression regressor is the best model for the prediction for used car prices.

11. FUTURE SCOPE:

Efficient use of deep learning such as LSTM (Long shortterm memory) or RNN (Recurrent Neural networks) can be implemented once enough data is collected. This can improve accuracy and decrease RMSE drastically. Currently, only few features are used to predict resale value of the car. This can be extended to more features. One can also implement CNN to determine physical condition of the car from images like identifying dents, scratches etc. and thus predicting more relevant resale value of a car.

12. APPENDIX:

SOURCE CODE:

Pre- processing the dataset:

```
import pandas as pd

import numpy as np

import matplotlib as plt

from sklearn.preprocessing import LabelEncoder

import pickle

df=pd.read_csv("autos.csv",header=0,sep=',',encoding='latin',)

df.columns

print(df.seller.value_counts())
```

```
df[df.seller != 'gewerblich']

df=df.drop('seller',1)

print(df.offerType.value_counts())

df[df.offerType != 'Gesuch']

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

print(df.shape)

 df = df[(df.powerPS > 50) & (df.powerPS <900)]

 print(df.shape)

 print(df.shape)

 df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]

 print(df.shape)

  df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen','postalCode', 'dateCreated'], axis='columns', inplace=True)

new_df = df.copy()

df.columns

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', 'kombi', '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")

labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']

mapper = {}

for i in labels:

  mapper[i] = LabelEncoder()

  mapper[i].fit(new_df[i])

  tr = mapper[i].transform(new_df[i])

  np.save(str('classes'+i+'.npy'), mapper[i].classes_)

  print(i,":", mapper[i])

  new_df.loc[:, i + '_labels'] = pd. Series (tr, index=new_df.index)


labeled = new_df[ ['price','yearOfRegistration' , 'powerPS' ,'kilometer' , 'monthOfRegistration']+
[x+"_labels" for x in labels]]

print(labeled.columns)

Y=labeled.iloc[:,0].values

X=labeled.iloc[:,1:].values
```

Y=Y.reshape(-1,1)

CHECK THE METRICS OF THE MODEL AND SAVE THE MODEL:

```
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


#regression model

from lightgbm import LGBMRegressor

data = pd.read_csv('autos_preprocessed.csv')

data.head()

labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']


mapper = {}

for i in labels:

    mapper[i] = LabelEncoder()

    mapper[i].fit(data[i])

    tr = mapper[i].transform(data[i])

    np.save(str('classes'+i+'.npy'), mapper[i].classes_)
```

```
    data.loc[:, i+'_labels'] = pd.Series(tr, index=data.index)


labeled = data[['price', 'yearOfRegistration','powerPS','kilometer','monthOfRegistration']
               +[x+"_labels" for x in labels]]


print(labeled.columns)
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
X = labeled.iloc[:,1:].values
Y = 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_estimators=300,objective="root_mean_squared_error",random_state=42,reg_sqrt=True)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
find_scores(Y_test, Y_pred, X_train)
pickle.dump(model, open('resale_model.sav', 'wb'))
```

PYTHON CODE FOR INTEGRATING WITH FLASK:

```
# 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
import requests
```

```
API_KEY = "UQfAKmX7EEgGGwkOyrDaKbtHjMUmz0teu62u6Rq27rVx"

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__)#initiate flask app

def load_model(file='../Result/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('../Result/'+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)
    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"fields": [['yearOfReg', 'powerPS', 'kilometer', 'monthOfRegistration','gearbox_labels', 'notRepairedDamage_labels', 'model_labels','brand_labels', 'fuelType_labels', 'vehicletype_labels']], "values": X}]}
    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/7f67cbed-6222-413b-9901-b2a72807ac82/predictions?version=2022-10-30', json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})
    predictions = response_scoring.json()
    print(response_scoring.json())
    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(host='localhost', debug=True, threaded=False)
```

13. GITHUB LINK & VIDEO URL:

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-9485-1659011635

VIDEO URL LINK:

(GOOGLE DRIVE LINK - VIDEO)

https://drive.google.com/file/d/1fwIR-FbPK6yT0rzQ7fNTOU76EfSk-hZi/view?usp=sharing

(GITHUB LINK- VIDEO)

https://github.com/IBM-EPBL/IBM-Project-9485-1659011635/tree/main/Final%20Deliverables/DEMO%20VIDEO