



M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.



A Project Report
On
CAR RESALE VALUE PREDICTION
Submitted in partial fulfillment for the award of the degree
of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of
Mrs.K.Makanyadevi,M.E.,
Assistant Professor/CSE

Submitted by

Team ID:PNT2022TMID15496

926719BCS4078 - MOHANASRIHARAN J

927619BCS4093 - PRATHAB S

927619BCS4109 - SARAN S

927619BCS4121 - VENIS C

**NAALAIYA THIRAN - EXPERIENTAL PROJECT BASED LEARNING
INITIATIVE**

**18CSE040L - PROFESSIONAL READINESS FOR INNOVATION,
EMPLOYABILITY ENTERPRENURSHIP**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(Autonomous)**

Karur 639 113

November,2022

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	4
2	LITERATURE SURVEY	6
3	IDEATION AND PROPOSED SOLUTION	8
4	REQUIREMENT ANALYSIS	14
5	PROJECT DESIGN	16
6	PROJECT PLANNING AND SCHEDULING	19
7	CODING AND SOLUTION	21
8	TESTING	29
9	RESULT	31
10	ADVANTAGES AND APPLICATIONS	32
11	CONCLUSION	33
12	FUTURE SCOPE	34
13	APPENDIX	35

TABLE OF FIGURE

CHAPTER NO	TITLE	PAGE NO
3.1	EMPATHY MAP	8
3.2	STEP 1	9
3.3	STEP 2	10
3.4	IDEA PRIORITIZATION	11
3.5	PROBLEM SOLUTION FIT	12
5.1	DATA FLOW DIAGRAM	16
5.2	SOLUTION AND TECHNICAL ARCHITECTURE	17
8.1	HOME PAGE	28
9.1	RESULT	30

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

1. With difficult economic conditions, it is likely that sales of second-hand imported (reconditioned) cars and used cars will increase. In many developed countries, it is common to lease a car rather than buying it outright. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e. its expected resale value. Thus, it is of commercial interest to sellers/financers to be able to predict the salvage value (residual value) of cars with accuracy.
2. In order to predict the resale value of the car, we proposed an intelligent, flexible, and effective system that is based on using regression algorithms. Considering the main factors which would affect the resale value of a vehicle a regression model is to be built that would give the nearest resale value of the vehicle. We will be using various regression algorithms and algorithm with the best accuracy will be taken as a solution, then it will be integrated to the web-based application where the user is notified with the status of his product.

1.2 PURPOSE

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

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

1. Several studies and related works have been done previously to predict used car prices around the world using different methodologies and approaches, with varying results of accuracy from 50% to 90%. In (Pudaruth, 2014) the researcher proposed to predict used car prices in Mauritius, where he applied different machine learning techniques to achieve his results like decision tree, K-nearest neighbours, Multiple Regression and Naïve Bayes algorithms to predict the used cars prices, based on historical data gathered from the newspaper. Achieved results ranged from accuracy of 60-70 percent, the author suggested using more sophisticated models and algorithms to make the evaluation, with the main weakness off the decision tree and naïve Bayes that it is required to discretize the price and classify it which accrue to more inaccuracies. Moreover, he suggested a larger set of data of data to train the models hence the data gathered was not sufficient. Gathered data from a German e-commerce site that totalled to 304,133 rows and 11 attributes to predict the prices of used car using different techniques and measured their results using Mean Absolute Error (MEA) to compare their results.

2.2 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

1. Brand or Type of the car one prefers like Ford, Hyundai
2. Model of the car namely Ford Figo, Hyundai Creta
3. Location like Delhi, Chennai, Mumbai
4. Year of manufacturing like 2020, 2021
5. Type of fuel namely Petrol, Diesel
6. Price range or Budget
7. 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.

2.3 REFERENCE

1. <https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=12220&context=theses>
2. <https://www.irjet.net/archives/V8/i4/IRJET-V8I4278.pdf>
3. <https://towardsdatascience.com/predicting-used-car-prices-with-machine-learning-techniques-8a9d8313952>
4. <https://www.enjoyalgorithms.com/blog/car-resale-value-predictor-using-random-forest-regressor>
5. http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26612934.pdf

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

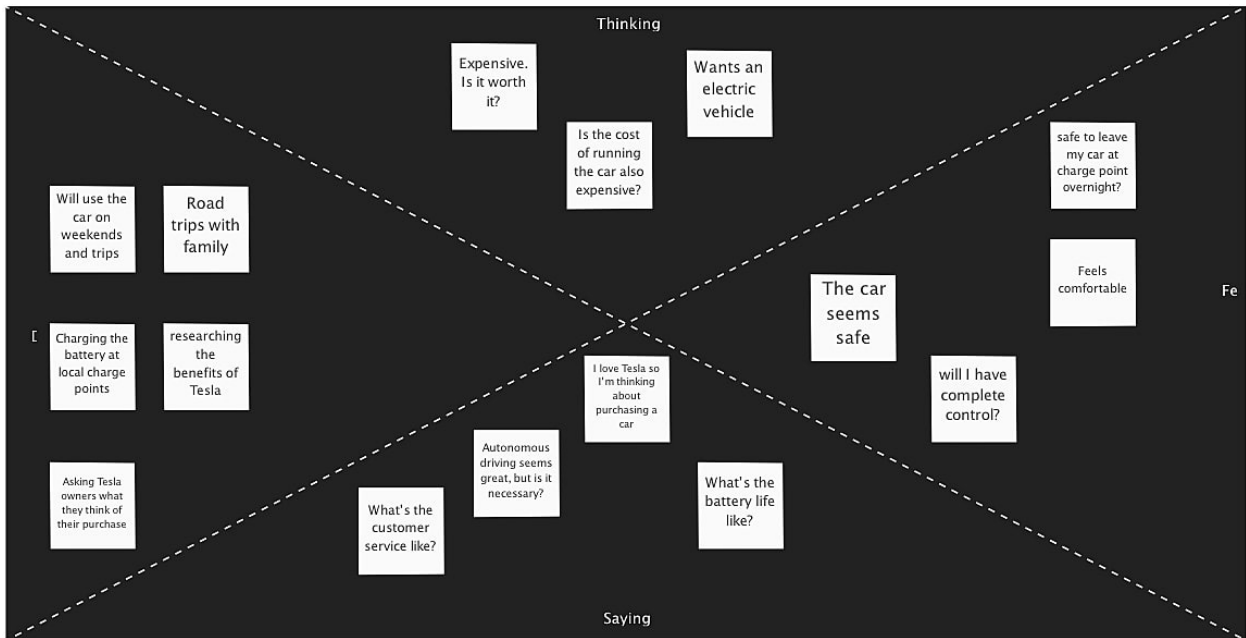


Figure 3.1 Empathy map

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Empathy mapping is a simple workshop activity that can be done with stakeholders, marketing and sales, product development, or creative teams to build empathy for end users. For teams involved in the design and engineering of products, services, or experiences, an empathy mapping session is a great exercise for groups to "get inside the heads" of users.

3.2 BRAINSTORM & IDEA PRIORITIZATION

Step 1: Team Gathering, Collaboration and Select the Problem statement

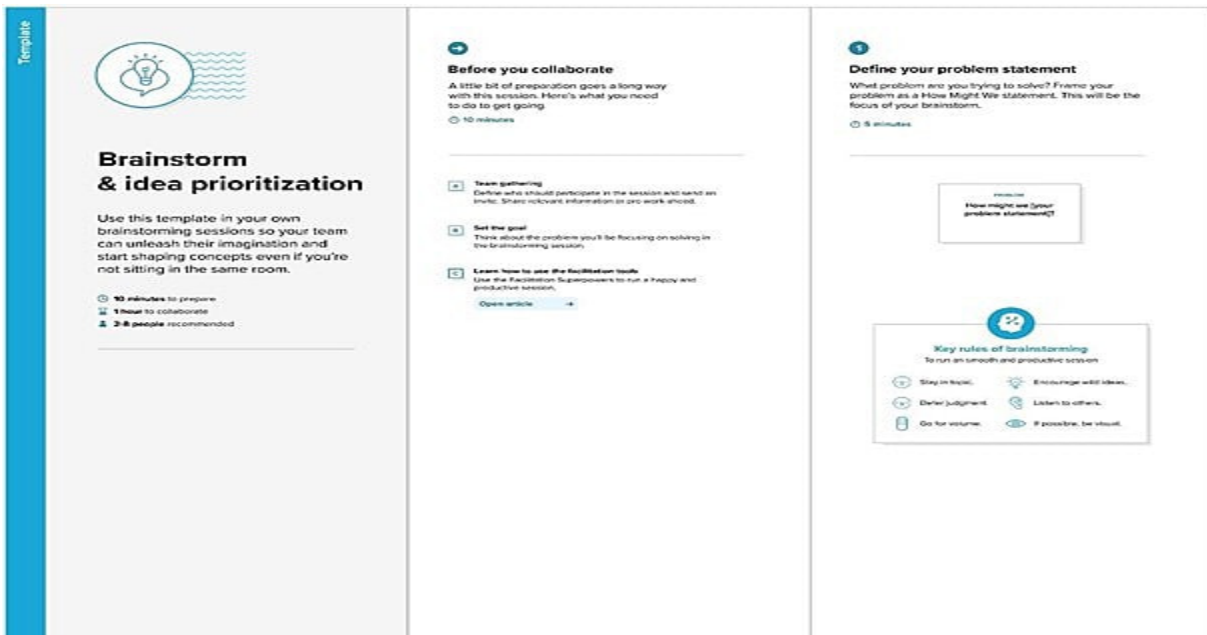


Figure 3.2 Team Gathering, Collaboration and Select the Problem statement

Brainstorming is a process where team members can pool together their ideas to find solutions to business problems. It gives us insights during data gathering as well as planning. Brainstorming in Project Management helps generate multiple ideas. We can reach a consensus on a specific solution from these ideas. There are 3 major types of brainstorming techniques. You can check out our guide to choose the best one.

Step 2: Brainstorm, Idea listing, and Grouping

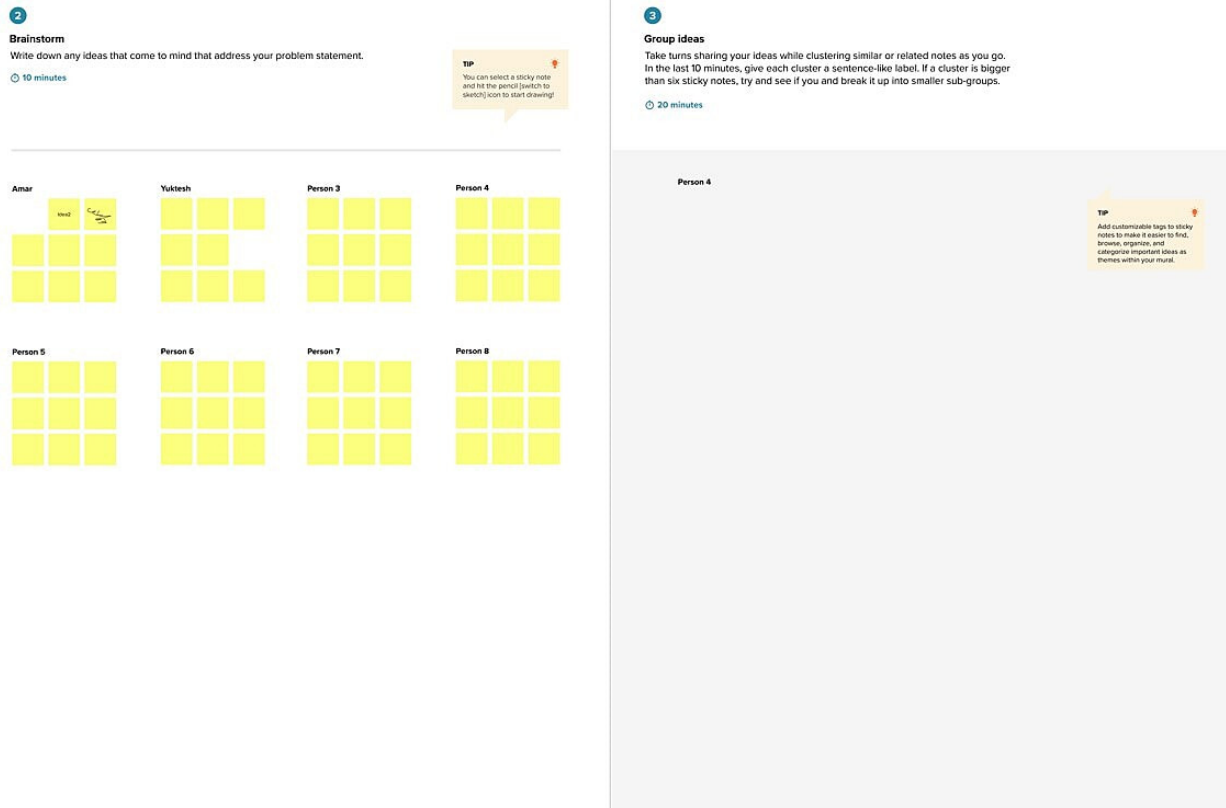


Figure 3.3 Brainstorm, Idea listing, and Grouping

1. One by one, participants share their views on each one of them.
2. The team repeats the process to reduce the number of ideas.
3. Then, the team selects the most suitable idea or ideas in the end.
4. Alternatively, the facilitator may take a step forward. They can use the NGT method. As we have discussed earlier, NGT is better than brainstorming. We can even use NGT across multiple groups.

Step 3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 20 minutes

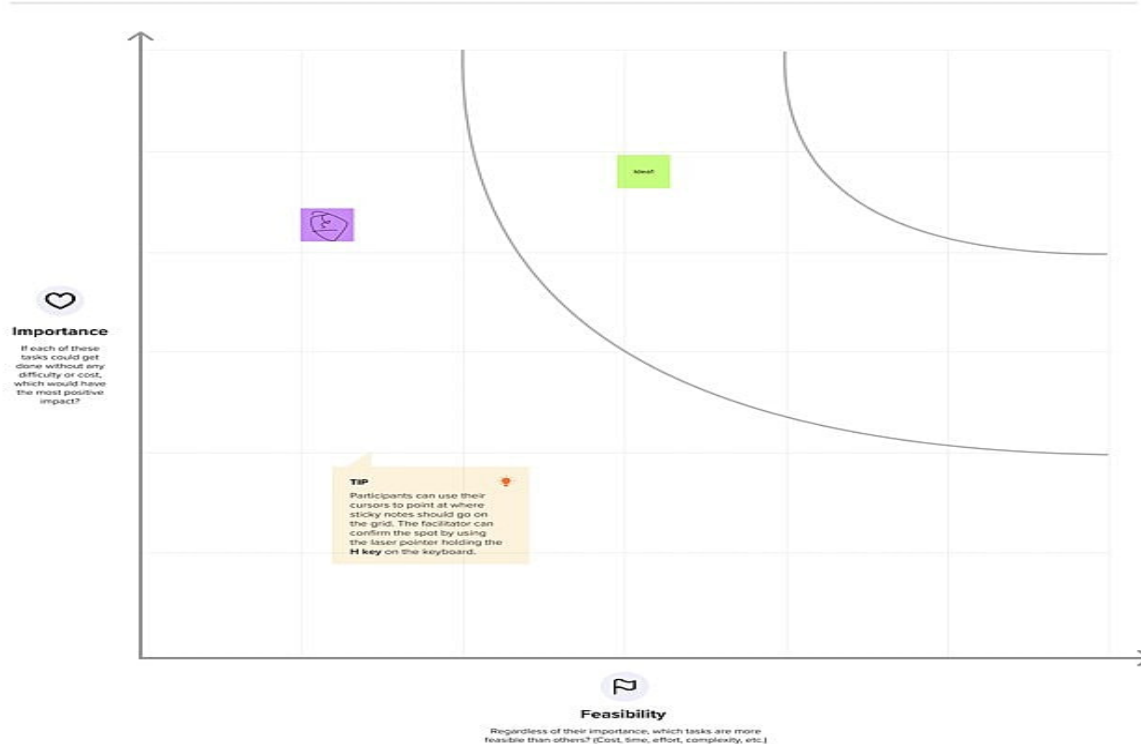


Figure 3.4 Idea prioritization

Idea prioritization is just a part of the idea management process. Having a structured idea management process and a systematic way of gathering, evaluating and prioritizing new ideas takes time. To make it work, the entire idea management process should be integrated to the everyday ways of working.

3.3 PROBLEM SOLUTION FIT

Project Title: Car resale value prediction

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID15486

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? Car buyers are my customers <div>CS</div>	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? spending power, budget, no cash, available cars <div>CC</div>	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem? or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? pen and paper is an alternative to digital notetaking <div>AS</div>	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. <div>JB</div>	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations <div>RC</div>	7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and <div>BE</div>	
Focus on JB, fit into RC				Focus on JB, fit into BE, understand RC

3. TRIGGERS What triggers customers to act? I seeing their neighbour buying cars, reading about a more efficient solution in the news. <div>TR</div>	10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business <div>SL</div>	8. CHANNELS of BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Extract online channels <div>CH</div>
--	---	---

Figure 3.5 Problem solution fit

3.4 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem statement (problem to be solved)	To predict the price of the used cars approximately using the dataset.
2.	Idea/solution description	One of the solutions of the problem is to identify the cars' mileage and usage of the car and predict the price.
3.	Novelty / uniqueness	This application can suggest a good price for the cars by giving the specifications of the car.
4.	Social impact/customer satisfaction	It helps the people by identifying the perfect car in the early stage and getting the quality and cost of the car.
5.	Business model (revenue model)	The application predicts the price of the car.
6.	Scalability of the solution	This application can be improved by introducing it in websites all over the world.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login via Email Login via password
FR-4	Car registration	Registering the car details
FR-5	Value Prediction	Predicting the car resale value

4.2 NON-FUNCTIONAL REQUIREMENTS

Following are the non-functional requirement of the proposed solution

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Predicting the resale value
NFR-2	Security	Providing security to the website
NFR-3	Reliability	Providing high reliability by predicting values for different types of cars
NFR-4	Performance	Providing high performance by using some machine learning techniques
NFR-5	Availability	It is used for all types of cars
NFR-6	Scalability	Predicting values for different types of cars

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

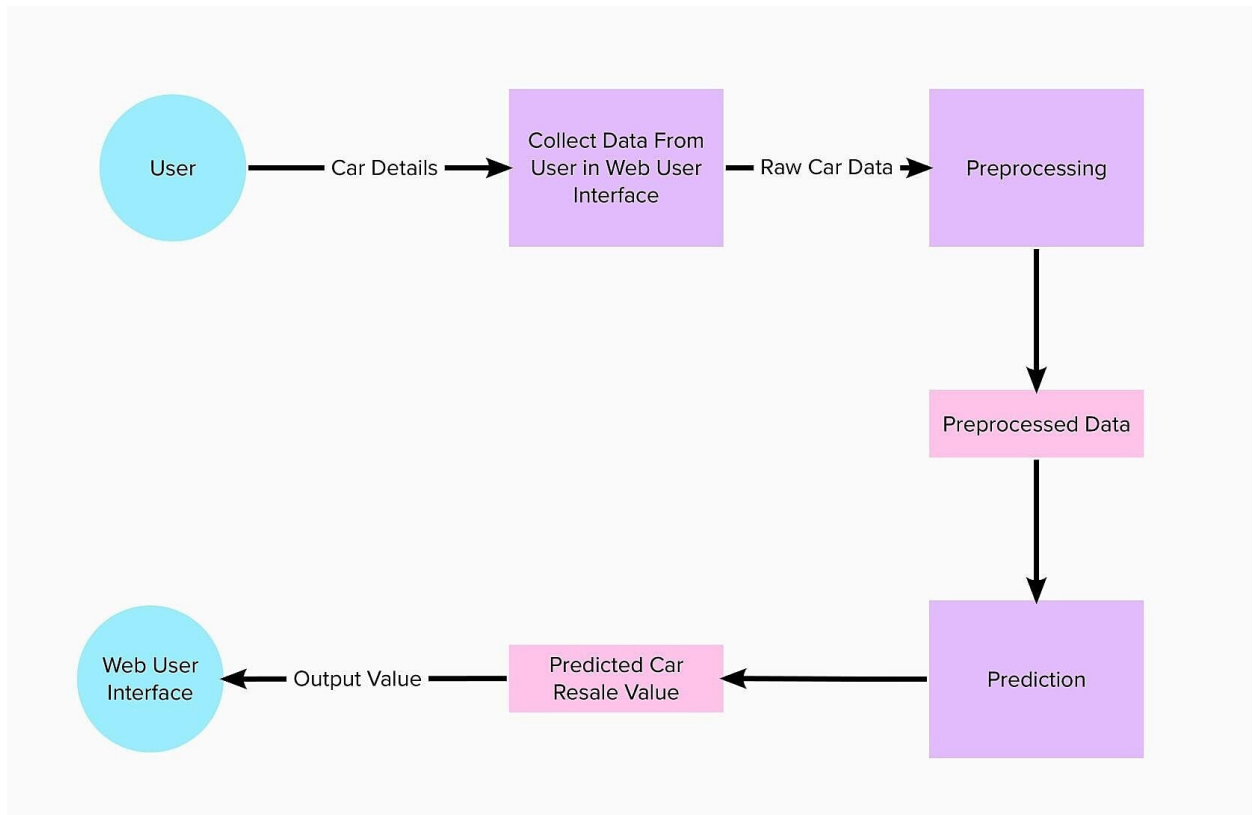


Figure 5.1 Data flow diagram

The Data Flow Diagram is a visual representation of the working of project from the user interaction to final output given for the user. Here the diagram describes the flow from the login to final output displayed.

5.2 SOLUTION & TECHNICAL ARCHITECTURE

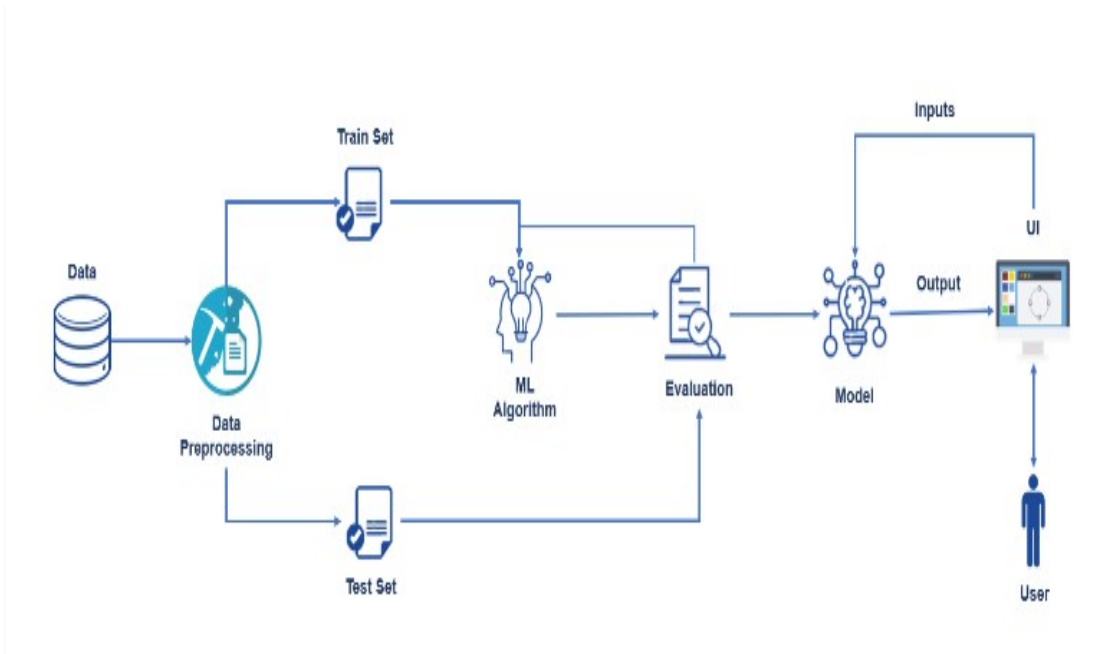


Figure 5.2 Solution and technical architecture

The technical architecture is diagrammatic representation which gives details of the building blocks involved in the project which is mostly on the technical details. Here the technical architecture describes the working of project which is based on machine learning model. The ML model is used to predict the performance of the vehicle.

5.3 USER STORIES

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

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

To accomplish the above task, you must complete the below activities and tasks:

1. Download the dataset.
2. Classify the dataset into train and test sets.
3. Import the suitable model
4. Load the trained data and fit the model.
5. Test the model.
6. Save the model and its dependencies.
7. Build a Web application using a flask that integrates with the model built.

6.2 SPRINT DELIVERY SCHEDULE

The delivery plan of project deliverables is a strategic element for every Project Manager. The goal of every project is, in fact, to produce a result that serves a specific purpose. With the word “Purpose”, we can mean the most disparate goals: a software program, a chair, a building, a translation, etc.... This Delivery plan helps to understand the process and Work Flow of the Project working by the Team Mates. Every Single Module is assigned to the teammates to showcase their work and contribution to developing the Project.

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed	Spring Release Date
Sprint– 1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint– 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint– 3	20	6 Days	07 Nov 2022	12 Nov 2022	15	13 Nov 2022
Sprint - 4	20	6 days	14 Nov 2022	19 Nov 2022	25	19 Nov 2022

CHAPTER 7

CODING AND SOLUTION

Google colab code

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, preprocessing, svm
from sklearn.preprocessing import StandardScaler, Normalizer
import math
import matplotlib
import seaborn as sns

%matplotlib inline
def category_values(dataframe, categories):
    for c in categories:
        print("\n", dataframe.groupby(by=c)[c].count().sort_values(ascending=False))
        print('Nulls: ', dataframe[c].isnull().sum())

def plot_correlation_map( df ):
    corr = df.corr()
    _, ax = plt.subplots( figsize =( 12 , 10 ) )
    cmap = sns.diverging_palette( 220 , 10 , as_cmap = True )
    _ = sns.heatmap(
        corr,
        cmap = cmap,
        square=True,
        cbar_kws={ 'shrink' : .9 },
        ax=ax,
        annot = True,
        annot_kws = { 'fontsize' : 12 }
    )
df = pd.read_csv('/content/drive/MyDrive/Imarticus/autos.csv', sep=',', header=0, encoding='cp1252')
#df = pd.read_csv('autos.csv.gz', sep=',', header=0, compression='gzip',encoding='cp1252')
df.sample(10)
```

```

df.describe()

print(df.seller.unique())
print(df.offerType.unique())
print(df.abtest.unique())
print(df.nrofPictures.unique())
df.drop(['seller', 'offerType', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode', 'dateCreated'],
axis='columns', inplace=True)

print("Too new: %d" % df.loc[df.yearOfRegistration >= 2017].count()['name'])
print("Too old: %d" % df.loc[df.yearOfRegistration < 1950].count()['name'])
print("Too cheap: %d" % df.loc[df.price < 100].count()['name'])
print("Too expensive: ", df.loc[df.price > 150000].count()['name'])
print("Too few km: ", df.loc[df.kilometer < 5000].count()['name'])
print("Too many km: ", df.loc[df.kilometer > 200000].count()['name'])
print("Too few PS: ", df.loc[df.powerPS < 10].count()['name'])
print("Too many PS: ", df.loc[df.powerPS > 500].count()['name'])
print("Fuel types: ", df['fuelType'].unique())
print("Damages: ", df['notRepairedDamage'].unique())
#print("Pics: ", df['nrOfPictures'].unique()) # nrOfPictures : number of pictures in the ad (unfortunately this field
contains everywhere a 0 and is thus useless (bug in crawler!) )
print("Vehicle types: ", df['vehicleType'].unique())
print("Brands: ", df['brand'].unique())

# Cleaning data
#valid_models = df.dropna()
#### Removing the duplicates
dedups = df.drop_duplicates(['name','price','vehicleType','yearOfRegistration'
                             , 'gearbox','powerPS','model','kilometer','monthOfRegistration','fuelType'
                             , 'notRepairedDamage'])
#### Removing the outliers
#### Removing the outliers
dedups = dedups[
    (dedups.yearOfRegistration <= 2016)
    & (dedups.yearOfRegistration >= 1950)
    & (dedups.price >= 100)
    & (dedups.price <= 150000)
    & (dedups.powerPS >= 10)
    & (dedups.powerPS <= 500)]

print("-----\nData kept for analysis: %d percent of the entire set\n-----" % (100 *
dedups['name'].count() / df['name'].count()))
dedups.isnull().sum()
dedups['notRepairedDamage'].fillna(value='not-declared', inplace=True)
dedups['fuelType'].fillna(value='not-declared', inplace=True)

```

```

dedups['gearbox'].fillna(value='not-declared', inplace=True)
dedups['vehicleType'].fillna(value='not-declared', inplace=True)
dedups['model'].fillna(value='not-declared', inplace=True)
dedups['namelen'] = [min(70, len(n)) for n in dedups['name']]
ax = sns.jointplot(x='namelen',
                  y='price',
                  data=dedups[['namelen','price']],
#                  data=dedups[['namelen','price']][dedups['model']=='golf'],
                  alpha=0.1,
                  size=8)
labels = ['name', 'gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
les = {}

```

for l **in** labels:

```

    les[l] = preprocessing.LabelEncoder()
    les[l].fit(dedups[l])
    tr = les[l].transform(dedups[l])
    dedups.loc[:, l + '_feat'] = pd.Series(tr, index=dedups.index)
labeled = dedups[ ['price'
                  , 'yearOfRegistration'
                  , 'powerPS'
                  , 'kilometer'
                  , 'monthOfRegistration'
                  , 'namelen'
                  + [x+"_feat" for x in labels]]
len(labeled['name_feat'].unique()) / len(labeled['name_feat'])
labeled.drop(['name_feat'], axis='columns', inplace=True)
plot_correlation_map(labeled)
labeled.corr()
labeled.corr().loc[:, 'price'].abs().sort_values(ascending=False)[1:]
labeled.drop(['model_feat'], axis='columns', inplace=True)
labeled.drop(['brand_feat'], axis='columns', inplace=True)
labeled.drop(['vehicleType_feat'], axis='columns', inplace=True)
labeled.drop(['notRepairedDamage_feat'], axis='columns', inplace=True)
Y = labeled['price']
X = labeled.drop(['price'], axis='columns', inplace=False)
from sklearn.model_selection import cross_val_score, train_test_split

```

#Split into train and validation

```

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state = 3)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.model_selection import GridSearchCV

hr = HistGradientBoostingRegressor()

```

```

param_grid = { "loss" : ['squared_error']
               , "max_leaf_nodes" : [31]
               , "min_samples_leaf": [20]
               , "max_depth": [None]
               , "max_iter": [500]}

gs = GridSearchCV(estimator=hr, param_grid=param_grid, cv=2, n_jobs=-1, verbose=1)
gs = gs.fit(X_train, y_train)
print('Score: %.2f' % gs.score(X_test, y_test))

print(gs.best_score_)
print(gs.best_params_)

import pickle
pickle.dump(gs,open('histmodel.pkl','wb'))

```

FLASK CODE

app.py

```

from flask import Flask,render_template,request
import pickle
import numpy as np
app = Flask(__name__)
model=pickle.load(open("histmodel.pkl","rb"))
@app.route("/")
def home():
    return render_template("newindex.html")

@app.route('/submit',methods=["POST","GET"])
def prediction():
    if request.method=="POST":
        yearofRegistration=request.form["yearofRegistration"]
        powerPS=request.form["powerPS"]
        kilometer=request.form["kilometer"]
        monthofRegistration=request.form["monthofRegistration"]
        namelen=request.form["namelen"]
        gearbox_feat=request.form["gearbox_feat"]
        if gearbox_feat == "manuell":
            gearbox_feat =1
        elif gearbox_feat == "auto":
            gearbox_feat =0
        fuelType_feat = request.form["fuelType_feat"]
        if fuelType_feat=="petrol":
            fuelType_feat=1
        elif fuelType_feat=="benzin":

```



```

        fuelType_feat=2
    elif fuelType_feat=="diesel":
        fuelType_feat=3
    elif fuelType_feat=="lpg":
        fuelType_feat=4
    elif fuelType_feat=="andere":
        fuelType_feat=5
    elif fuelType_feat=="hybrid":
        fuelType_feat=6
    elif fuelType_feat=="cng":
        fuelType_feat=7
    elif fuelType_feat=="elektro":
        fuelType_feat=8
    int_features = [yearofRegistration,
powerPS,kilometer,monthofRegistration,namelen,gearbox_feat,fuelType_feat]
    features = [np.array(int_features, dtype=int)]
    prediction=model.predict(features)
    return render_template("newssubmit.html",prediction=round(prediction[0],2))

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

```

HTML CODE

newindex.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Car Price Prediction</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-

```

```

awesome.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<style>
  body{
    background-color: #f2f2f2;
  }
  .container{
    background-color: white;
    margin-top: 50px;
    padding: 20px;
    border-radius: 10px;
  }
  .form-group{
    margin-top: 20px;
  }
  .form-control{
    border-radius: 10px;
  }
  .btn{
    border-radius: 10px;
    margin-top: 20px;
  }
  .fa{
    font-size: 30px;
    color: #f2f2f2;
  }

</style>
</head>
<body>
  <div class="container">
    <h1 class="text-center">Car Price Prediction</h1>
    <form action="/submit" method="POST">
      <div class="form-group">
        <label for="yearofRegistration">Year of Registration</label>
        <input type="number" class="form-control" id="yearofRegistration" name="yearofRegistration"

```

```

placeholder="Enter Year of Registration">
    </div>
    <div class="form-group">
        <label for="powerPS">Power PS</label>
        <input type="number" class="form-control" id="powerPS" name="powerPS" placeholder="Enter Power
PS">
    </div>
    <div class="form-group">
        <label for="kilometer">Kilometer</label>
        <input type="number" class="form-control" id="kilometer" name="kilometer" placeholder="Enter
Kilometer">

</div>
    <div class="form-group">
        <label for="monthofRegistration">Month of Registration</label>
        <input type="number" class="form-control" id="monthofRegistration" name="monthofRegistration"
placeholder="Enter Month of Registration">
    </div>
    <div class="form-group">
        <label for="namelen">Name Length</label>
        <input type="number" class="form-control" id="namelen" name="namelen" placeholder="Enter Name
Length">
    </div>
    <div class="form-group">
        <label for="gearbox_feat">Gearbox</label>
        <select class="form-control" id="gearbox_feat" name="gearbox_feat">
            <option value="manuell">Manuell</option>
            <option value="auto">Auto</option>
        </select>
    </div>
    <div class="form-group">
        <label for="fuelType_feat">Fuel Type</label>
        <select class="form-control" id="fuelType_feat" name="fuelType_feat">
            <option value="andere">Andere</option>
            <option value="benzin">Benzin</option>
            <option value="cng">CNG</option>
            <option value="diesel">Diesel</option>
            <option value="elektro">Elektro</option>
            <option value="hybrid">Hybrid</option>
            <option value="lpg">LPG</option>
        </select>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>

```

```

    </div>
</body>
</html>

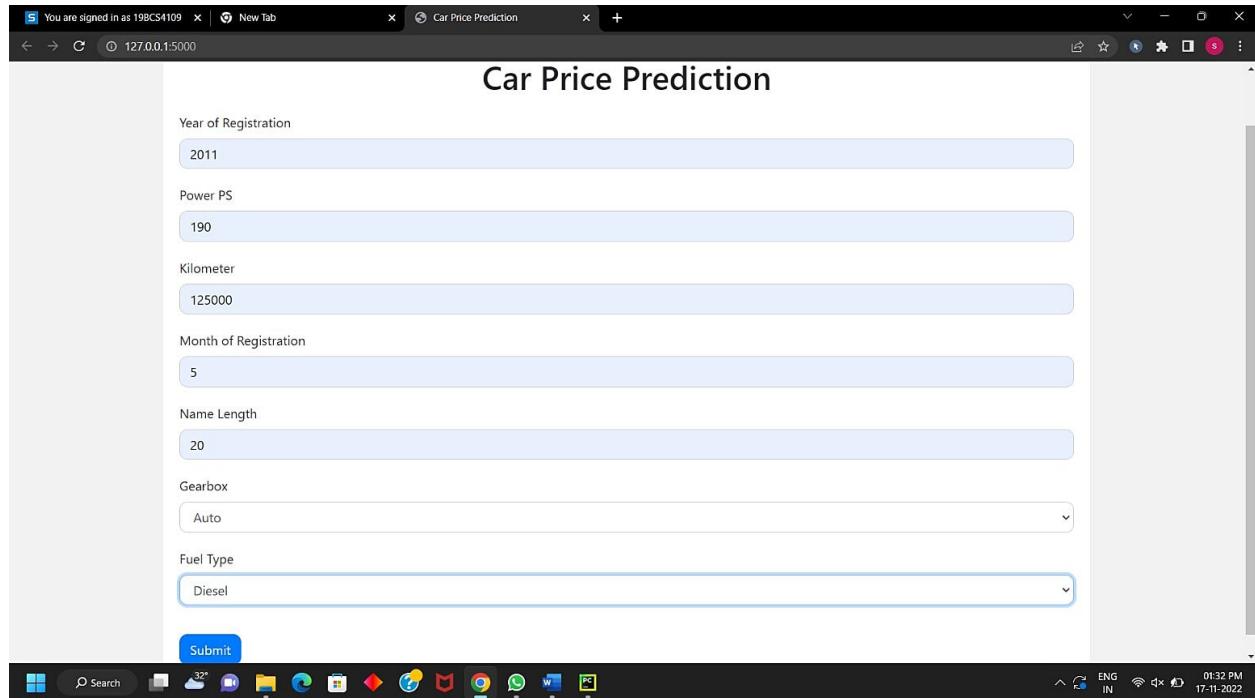
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Car Price Prediction</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-
awesome.min.css">
    <style>
        body{
            background-color: #f2f2f2;
        }
        .container{
            background-color: white;
            margin-top: 50px;
            padding: 20px;
            border-radius: 10px;
        }
        .form-group{
            margin-top: 20px;
        }
        .form-control{
            border-radius: 10px;
        }
        .btn{
            border-radius: 10px;
            margin-top: 20px;
        }
        .fa{
            font-size: 30px;
            color: #f2f2f2; }
    </style>
</head>
<body>
    <div class="container">
        <h1 class="text-center">Car Price Prediction</h1>
        <h3 class="text-center">Predicted Price: {{price}}</h3>
    </div>
</body>
</html>

```

CHAPTER 8

TESTING

TEST CASES



The screenshot shows a web browser window with the title "Car Price Prediction". The browser's address bar shows "127.0.0.1:5000". The page contains a form with the following fields and values:

- Year of Registration: 2011
- Power PS: 190
- Kilometer: 125000
- Month of Registration: 5
- Name Length: 20
- Gearbox: Auto (selected from a dropdown menu)
- Fuel Type: Diesel (selected from a dropdown menu)

A blue "Submit" button is located at the bottom of the form. The Windows taskbar is visible at the bottom of the screen, showing the time as 01:32 PM on 17-11-2022.

Figure: 8.1 Home Page

Testcase 1:

1. Enter the year of registration-2011
2. Enter the power of the engine-190
3. Enter the kilometers driven-125000
4. Enter the month of registration-5
5. Choose the gearbox type-Auto
6. Choose the fuel type-Diesel

Testcase 2:

1. Enter the year of registration-2012
2. Enter the power of the engine-100
3. Enter the kilometers driven-10000
4. Enter the month of registration-2
5. Choose the gearbox type-Auto
6. Choose the fuel type-Petrol

Testcase 3:

1. Enter the year of registration-2001
2. Enter the power of the engine-50
3. Enter the kilometers driven-23000
4. Enter the month of registration-1
5. Choose the gearbox type-Manuell
6. Choose the fuel type-Diesel

CHAPTER 9

RESULTS

The image displays two screenshots of a web browser. The top screenshot shows the 'Car Price Prediction' form with the following inputs: Year of Registration (2011), Power PS (190), Kilometer (125000), Month of Registration (5), Name Length (20), Gearbox (Auto), and Fuel Type (Diesel). A blue 'Submit' button is at the bottom. The bottom screenshot shows the result page with the title 'Car Price Prediction' and the text 'Predicted Price: 20647.75'.

Car Price Prediction

Year of Registration
2011

Power PS
190

Kilometer
125000

Month of Registration
5

Name Length
20

Gearbox
Auto

Fuel Type
Diesel

Submit

Car Price Prediction
Predicted Price: 20647.75

FIGURE:9.1 RESULT

CHAPTER 10

ADVANTAGES & APPLICATIONS

10.1 ADVANTAGES

1. The proposed model could predict the approximate price of the car.
2. Easy to use UI.
3. Model has some good accuracy in detecting the price just by taking the input (features).

10.2 APPLICATIONS

1. This web application can be used by the customers who need to buy second hand car in a dreamt specifications.
2. So our website helps them to sit in a spot and check the price of the desired dream car.

CHAPTER 11

CONCLUSION

An efficient machine learning model is built by training, testing, and evaluating three machine learning regressors named Random Forest Regressor, Linear Regression, and Bagging Regressor. As a result of pre-processing and transformation, Hist gradient boosting Regressor came out on top with 77% accuracy. Each experiment was performed in real-time within the Google Colab environment. In comparison to the system's integrated Jupyter notebook and Anaconda's platform, algorithms took less training time in Google Colab.

CHAPTER 12

FUTURE SCOPE

The price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to 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 .

CHAPTER 13

APPENDIX

SOURCE CODE

Data preprocessing and model building

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ] import pandas as pd

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, preprocessing, svm
from sklearn.preprocessing import StandardScaler, Normalizer
import math
import matplotlib
import seaborn as sns

%matplotlib inline
```

```
def category_values(dataframe, categories):
    for c in categories:
        print('\n', dataframe.groupby(by=c)[c].count().sort_values(ascending=False))
        print('Nulls: ', dataframe[c].isnull().sum())

def plot_correlation_map( df ):
    corr = df.corr()
    _, ax = plt.subplots( figsize = ( 12 , 10 ) )
    cmap = sns.diverging_palette( 220 , 10 , as_cmap = True )
    _ = sns.heatmap(
        corr,
        cmap = cmap,
        square=True,
        cbar_kws={ 'shrink' : .9 },
```

```
df = pd.read_csv('/content/drive/MyDrive/Imarticus/autos.csv', sep=',', header=0, encoding='cp1252')
#df = pd.read_csv('autos.csv.gz', sep=',', header=0, compression='gzip', encoding='cp1252')
df.sample(10)
```

	dateCrawled	name	seller	offerType	price	abtest
123322	2016-03-19 19:37:20	Nissan_Micra_K11_Rostschaden	privat	Angebot	250	control
10005	2016-03-24 16:50:54	BMW_318i	privat	Angebot	600	control
285021	2016-04-02 15:40:51	BMW_330d_Automatik_INDIVIDUAL_Sportpaket_Navi_...	privat	Angebot	11849	control
363821	2016-04-04 00:55:50	Mercedes_Benz_A_170_Avantgarde	privat	Angebot	5000	test
341766	2016-03-30 10:37:55	Bmw_e30_325i_touring	privat	Angebot	1650	test
306410	2016-03-27 19:47:44	Fiat_Panda_1.1_Active	privat	Angebot	1300	control
175659	2016-03-28 10:58:02	BMW_X3_3.0i_230_PS/24V_Xenon_Leder_Schiebe...	privat	Angebot	12000	control
175879	2016-03-05 11:00:40	Golf 3 tuev bis 09 2017 q kat gruene plakette ...	privat	Angebot	580	test

```
print(df.seller.unique())
print(df.offerType.unique())
print(df.abtest.unique())
print(df.nrOfPictures.unique())
```

```
['privat' 'gewerblich']
['Angebot' 'Gesuch']
['test' 'control']
[0]
```

```
[ ] df.drop(['seller', 'offerType', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode', 'dateCreated'],
```

```
[ ] print("Too new: %d" % df.loc[df.yearOfRegistration >= 2017].count()['name'])
print("Too old: %d" % df.loc[df.yearOfRegistration < 1950].count()['name'])
print("Too cheap: %d" % df.loc[df.price < 100].count()['name'])
print("Too expensive: " , df.loc[df.price > 150000].count()['name'])
print("Too few km: " , df.loc[df.kilometer < 5000].count()['name'])
print("Too many km: " , df.loc[df.kilometer > 200000].count()['name'])
print("Too few PS: " , df.loc[df.powerPS < 10].count()['name'])
print("Too many PS: " , df.loc[df.powerPS > 500].count()['name'])
print("Fuel types: " , df['fuelType'].unique())
print("Damages: " , df['notRepairedDamage'].unique())
#print("Pics: " , df['nrOfPictures'].unique()) # nrOfPictures : number of pictures in the ad (unfortunately this f
print("Vehicle types: " , df['vehicleType'].unique())
print("Brands: " , df['brand'].unique())
```

```
# Cleaning data
#valid_models = df.dropna()
```

```
#### Removing the duplicates
dedups = df.drop_duplicates(['name', 'price', 'vehicleType', 'yearOfRegistration'])
```

```

, 'notRepairedDamage'])

#### Removing the outliers
dedups = dedups[
    (dedups.yearOfRegistration <= 2016)
    & (dedups.yearOfRegistration >= 1950)
    & (dedups.price >= 100)
    & (dedups.price <= 150000)
    & (dedups.powerPS >= 10)
    & (dedups.powerPS <= 500)]

print("-----\nData kept for analisis: %d percent of the entire set\n-----")

```

↗ Too new: 14680
 Too old: 289
 Too cheap: 13320
 Too expensive: 232
 Too few km: 0
 Too many km: 0
 Too few PS: 41040
 Too many PS: 835
 Fuel types: ['benzin' 'diesel' nan 'lpg' 'andere' 'hybrid' 'cng' 'elektro']
 Damages: [nan 'ja' 'nein']
 Vehicle types: [nan 'coupe' 'suv' 'kleinwagen' 'limousine' 'cabrio' 'bus' 'kombi' 'andere']
 Brands: ['volkswagen' 'audi' 'jeep' 'skoda' 'bmw' 'peugeot' 'ford' 'mazda' 'nissan' 'renault' 'mercedes_benz' 'opel' 'seat' 'citroen' 'honda' 'fiat' 'mini' 'smart' 'hyundai' 'sonstige_autos' 'alfa_romeo' 'subaru' 'volvo' 'mitsubishi' 'kia' 'suzuki' 'lancia' 'porsche' 'toyota' 'chevrolet' 'dacia' 'daihatsu' 'trabant' 'saab' 'chrysler' 'jaguar' 'daewoo' 'rover' 'land_rover' 'lada']

 Data kept for analisis: 81 percent of the entire set

```
[ ] dedups.isnull().sum()
```

```

name                0
price               0
vehicleType         10818
yearOfRegistration  0
gearbox             5260
powerPS             0
model              11347
kilometer           0
monthOfRegistration  0
fuelType           15400
brand               0
notRepairedDamage   42124
dtype: int64

```

```
[ ] dedups['notRepairedDamage'].fillna(value='not-declared', inplace=True)
dedups['fuelType'].fillna(value='not-declared', inplace=True)
dedups['gearbox'].fillna(value='not-declared', inplace=True)
dedups['vehicleType'].fillna(value='not-declared', inplace=True)
dedups['model'].fillna(value='not-declared', inplace=True)

```

```

▶ dedups['namelen'] = [min(70, len(n)) for n in dedups['name']]

ax = sns.jointplot(x='namelen',
                  y='price',
                  data=dedups[['namelen', 'price']],
                  # data=dedups[['namelen', 'price']][dedups['model']=='golf'],
                  alpha=0.1,
                  size=8)

```

```

▶ labels = ['name', 'gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
les = {}

for l in labels:
    les[l] = preprocessing.LabelEncoder()
    les[l].fit(dedups[l])
    tr = les[l].transform(dedups[l])
    dedups.loc[:, l + '_feat'] = pd.Series(tr, index=dedups.index)

labeled = dedups[ ['price'
                  , 'yearOfRegistration'
                  , 'powerPS'
                  , 'kilometer'
                  , 'monthOfRegistration'
                  , 'namelen']
                  + [x + "_feat" for x in labels]]

[ ] len(labeled['name_feat'].unique()) / len(labeled['name_feat'])

0.6224184813880769

[ ] labeled.drop(['name_feat'], axis='columns', inplace=True)

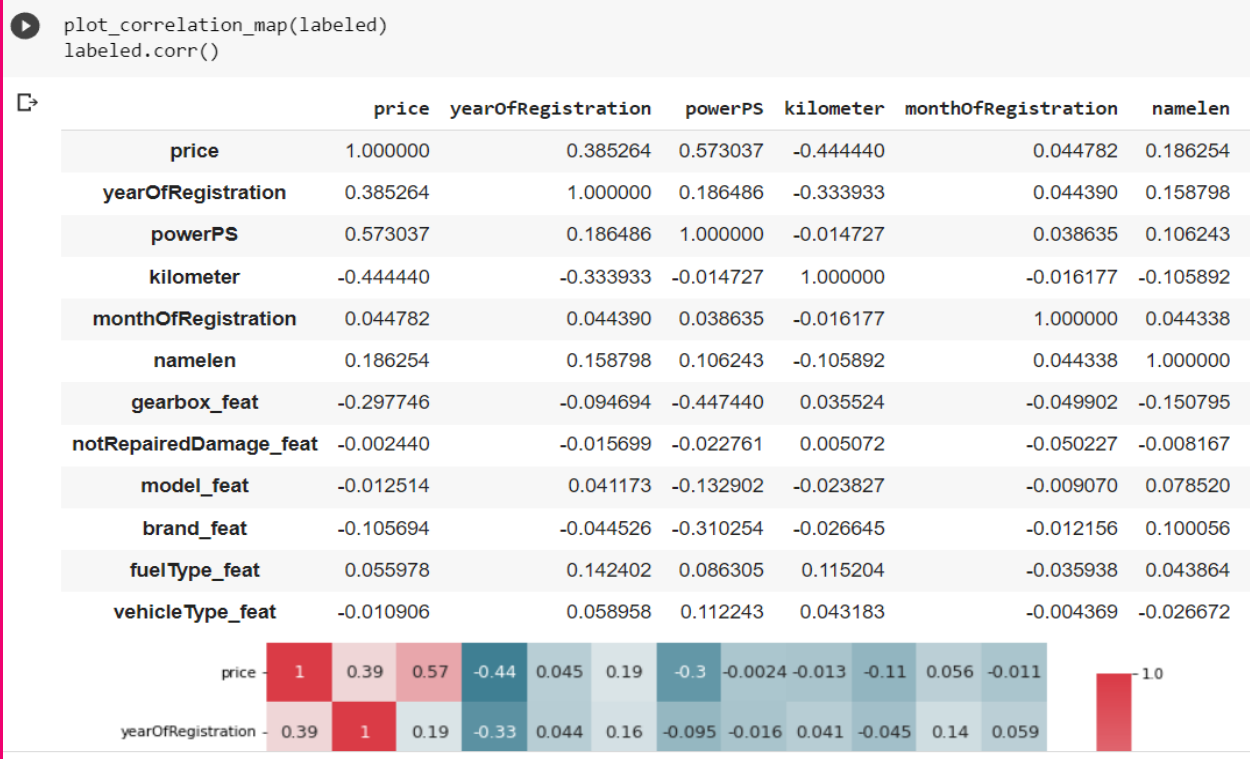
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/index.html#errors,

```

+

+



```
▶ labeled.corr().loc[:, 'price'].abs().sort_values(ascending=False)[1:]
```

```
↳ powerPS                0.573037  
kilometer                0.444440  
yearOfRegistration       0.385264  
gearbox_feat            0.297746  
namelen                 0.186254  
brand_feat              0.105694  
fuelType_feat           0.055978  
monthOfRegistration      0.044782  
model_feat              0.012514  
vehicleType_feat        0.010906  
notRepairedDamage_feat  0.002440  
Name: price, dtype: float64
```

```
[ ] labeled.drop(['model_feat'], axis='columns', inplace=True)  
labeled.drop(['brand_feat'], axis='columns', inplace=True)  
labeled.drop(['vehicleType_feat'], axis='columns', inplace=True)  
labeled.drop(['notRepairedDamage_feat'], axis='columns', inplace=True)
```

/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: 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/index.html#errors

```
[ ] Y = labeled['price']  
X = labeled.drop(['price'], axis='columns', inplace=False)
```

```
[ ] from sklearn.model_selection import cross_val_score, train_test_split  
  
#Split into train and validation  
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state = 3)  
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(203769, 7) (100364, 7) (203769,) (100364,)
```

```
▶ from sklearn.ensemble import HistGradientBoostingRegressor  
from sklearn.model_selection import GridSearchCV  
  
hr = HistGradientBoostingRegressor()  
  
param_grid = { "loss" : ['squared_error']  
               , "max_leaf_nodes" : [31]  
               , "min_samples_leaf": [20]  
               , "max_depth": [None]  
               , "max_iter": [500]}  
  
gs = GridSearchCV(estimator=hr, param_grid=param_grid, cv=2, n_jobs=-1, verbose=1)  
gs = gs.fit(X_train, y_train)  
print('Score: %.2f' % gs.score(X_test, y_test))
```

```
↳ Fitting 2 folds for each of 1 candidates, totalling 2 fits  
Score: 0.78
```

```
[ ] print(gs.best_score_)  
print(gs.best_params_)
```

```
0.7742344194771651
```

```
[ ] import pickle  
    pickle.dump(gs,open('histmodel.pkl','wb'))
```


FLASK DEPLOYMENT

```
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn.ensemble._forest
app = Flask(__name__)
model=pickle.load(open("histmodel.pkl", "rb"))
@app.route("/")
def home():
    return render_template("newindex.html")

@app.route('/submit', methods=["POST", "GET"])
def prediction():
    if request.method=="POST":
        yearofRegistration=request.form["yearofRegistration"]
        powerPS=request.form["powerPS"]
        kilometer=request.form["kilometer"]
        monthofRegistration=request.form["monthofRegistration"]
        namelen=request.form["namelen"]
        gearbox_feat=request.form["gearbox_feat"]
        if gearbox_feat == "manuell":
            gearbox_feat =1
        elif gearbox_feat == "auto":
            gearbox_feat =0
        fuelType_feat = request.form["fuelType_feat"]
        if fuelType_feat=="petrol":
            fuelType_feat=1
        elif fuelType_feat=="benzin":
            fuelType_feat=2
        elif fuelType_feat=="diesel":
            fuelType_feat=3
        elif fuelType_feat=="lgp":
            fuelType_feat=4
```

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

HTML PAGE FOR USER INTERFACE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Car Price Prediction</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384"
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
body{
    background-color: #f2f2f2;
}
.container{
    background-color: white;
    margin-top: 50px;
    padding: 20px;
    border-radius: 10px;
}
.form-group{
    margin-top: 20px;
}
.form-control{
    border-radius: 10px;
```

```
.form-control{
  border-radius: 10px;
}
.btn{
  border-radius: 10px;
  margin-top: 20px;
}
.fa{
  font-size: 30px;
  color: #f2f2f2;
}

</style>
</head>
<body>
  <div class="container">
    <h1 class="text-center">Car Price Prediction</h1>
    <form action="/submit" method="POST">
      <div class="form-group">
        <label for="yearofRegistration">Year of Registration</label>
        <input type="number" class="form-control" id="yearofRegistration" name="yearofRegistration" placeholder="Enter Year of Registration">
      </div>
      <div class="form-group">
        <label for="powerPS">Power PS</label>
        <input type="number" class="form-control" id="powerPS" name="powerPS" placeholder="Enter Power PS">
      </div>
      <div class="form-group">
        <label for="kilometer">Kilometer</label>
        <input type="number" class="form-control" id="kilometer" name="kilometer" placeholder="Enter Kilometer">
      </div>
      <div class="form-group">
```

```

        <input type="number" class="form-control" id="monthofRegistration" name="monthofRegistration" placeholder="Enter Month of Registration" />
    </div>
    <div class="form-group">
        <label for="namelen">Name Length</label>
        <input type="number" class="form-control" id="namelen" name="namelen" placeholder="Enter Name Length" />
    </div>
    <div class="form-group">
        <label for="gearbox_feat">Gearbox</label>
        <select class="form-control" id="gearbox_feat" name="gearbox_feat">
            <option value="manuell">Manuell</option>
            <option value="auto">Auto</option>
        </select>
    </div>
    <div class="form-group">
        <label for="fuelType_feat">Fuel Type</label>
        <select class="form-control" id="fuelType_feat" name="fuelType_feat">
            <option value="andere">Andere</option>
            <option value="benzin">Benzin</option>
            <option value="cng">CNG</option>
            <option value="diesel">Diesel</option>
            <option value="elektro">Elektro</option>
            <option value="hybrid">Hybrid</option>
            <option value="lpg">LPG</option>
        </select>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
</div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Car Price Prediction</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha"
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

  <style>
    body{
      background-color: #f2f2f2;
    }
    .container{
      background-color: white;
      margin-top: 50px;
      padding: 20px;
      border-radius: 10px;
    }
    .form-group{
      margin-top: 20px;
    }
    .form-control{
      border-radius: 10px;
    }
    .btn{
      border-radius: 10px;
      margin-top: 20px;
    }
    .fa{
      font-size: 30px;
      color: #f2f2f2;
    }
  </style>

```

```
        border-radius: 10px;
    }
    .btn{
        border-radius: 10px;
        margin-top: 20px;
    }
    .fa{
        font-size: 30px;
        color: #f2f2f2;
    }
</style>
</head>
<body>
    <div class="container">
        <h1 class="text-center">Car Price Prediction</h1>
        <h3 class="text-center">Predicted Price: {{price}}</h3>
    </div>
</body>
</html>
```

GIT HUB LINK: <https://github.com/IBM-EPBL/IBM-Project-20664-1659759860>

PROJECT DEMO LINK: https://drive.google.com/file/d/1_sQ1gxph7syvth-KlC6hNujSwpVL4dif/view?usp=share_link