PROJECT REPORT

TEAM ID:PNT2022TMID53078
Project Name: Car Resale Value Prediction
Team Members:
Mukund Balaji
Shravan S
Shreesh S
Saisrinath N

1. **INTRODUCTION**
   1.1 Project Overview
   - To understand the problem to classify if it is a regression or a classification kind of problem.
   - To pre-process/clean the data using different data pre-processing techniques.
   - Applying different algorithms according to the dataset

   1.2 Purpose
   To build a working web application using the Python Flask Framework and deploy our built model on it to meet user satisfaction.

2. **LITERATURE SURVEY**
   2.1 Existing problem
   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.
   2.2 References

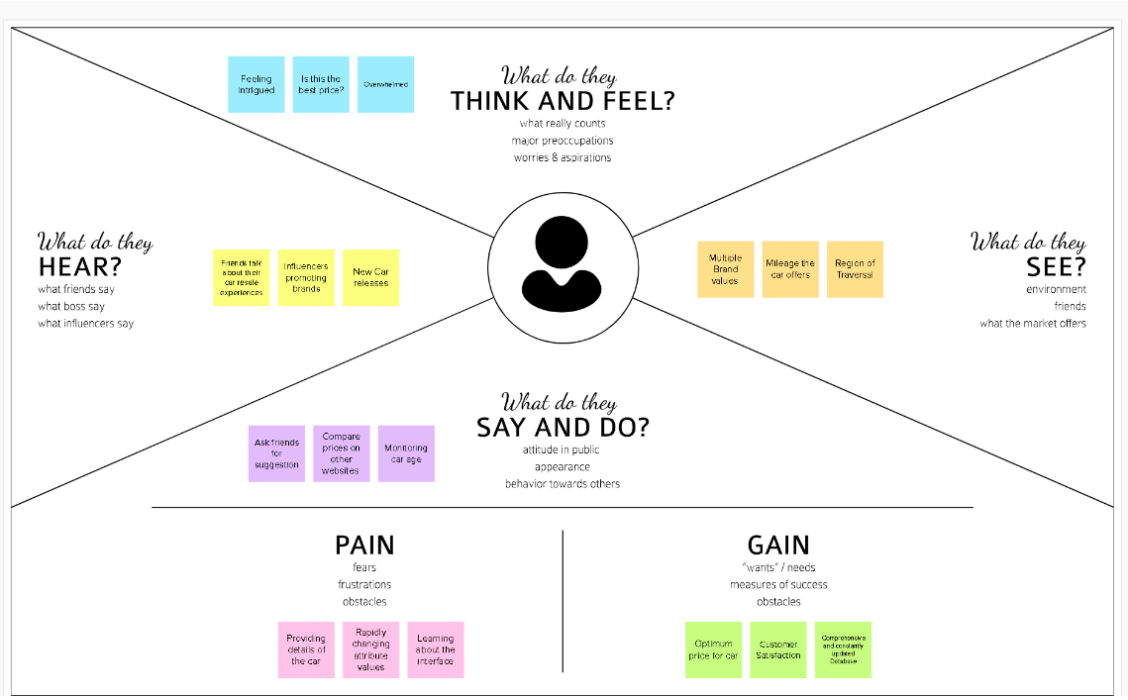
   - Voß, S. (2013). Resale Price Prediction in the Used Car Market.
   - Kiran, S., 2020. Prediction of resale value of the car using linear regression algorithm. *Int. J. Innov. Sci. Res. Technol*, *6*(7), pp.382-386.\
   - Gegic, E., Isakovic, B., Keco, D., Masetic, Z. and Kevric, J., 2019. Car price prediction using machine learning techniques. *TEM Journal*, *8*(1), p.113.
   - Ganesh, Mukkesh & Venkatasubbu, Pattabiraman. (2019). Used Cars Price Prediction using Supervised Learning Techniques. International Journal of Engineering and Advanced Technology. 9. 216-223. 10.35940/ijeat.A1042.1291S319.
   - Pudaruth, S., 2014. Predicting the price of used cars using machine learning techniques. *Int. J. Inf. Comput. Technol*, *4*(7), pp.753-764.

   2.3 Problem Statement Definition

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.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Team lead

Establish Parameters

Fit regression curves

Visualization Plots for parameters vs price

Try Ensembling of models

Member 1

Eliminate irrelevant parameters

Include consensus opinion

Exclude outliers

K-fold cross validation testing

Member 2

Use web-scraping to extract datasets

Prerprocess data

Use IBM Watson

Use React.js

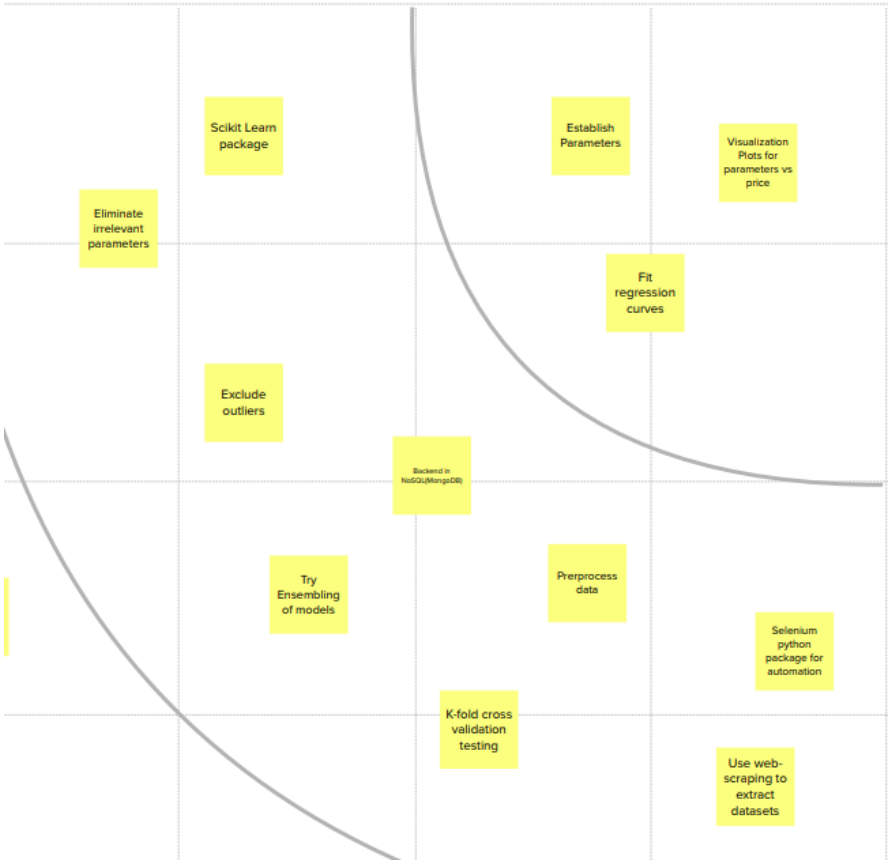
Member 3

Selenium python package for automation

Backend in NoSQL(MongoDB)

Scikit Learn package

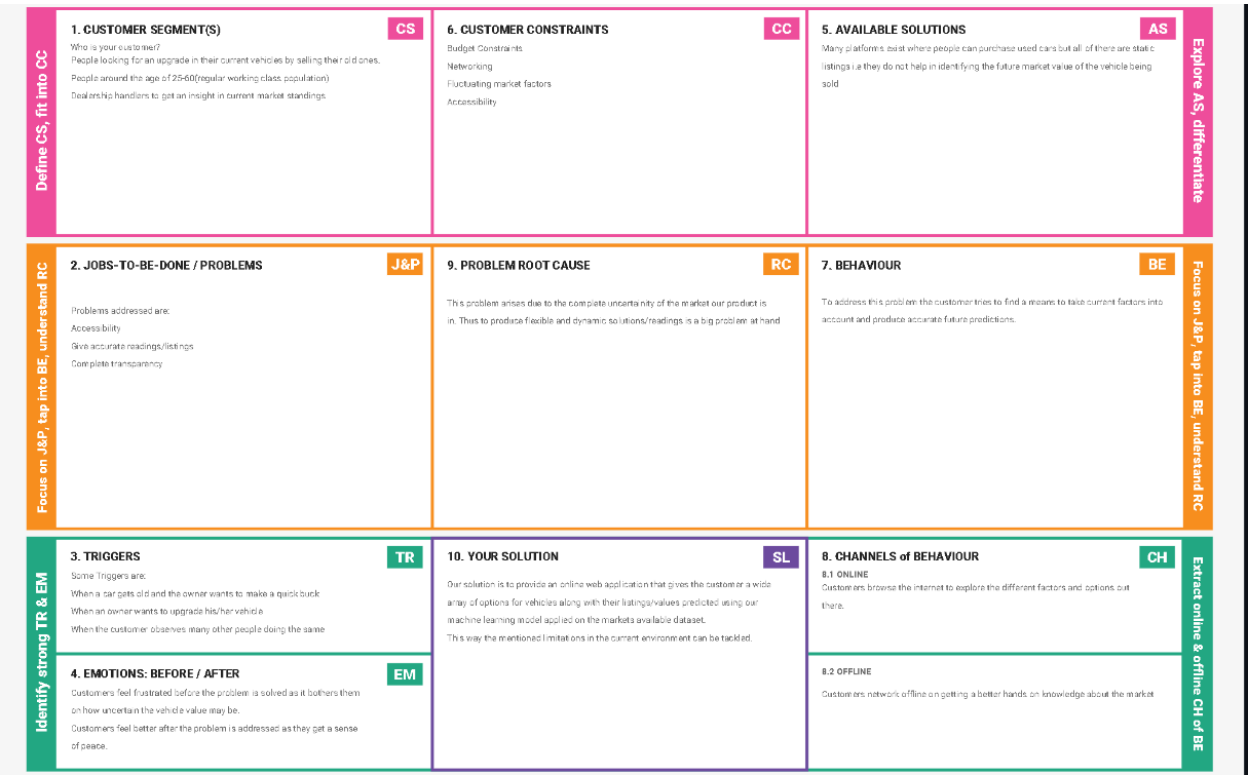
Keep environment factors in mind



3.3 Proposed Solution

The dataset has to be pre-processed in an appropriate way prior to Machine Learning model implementation. The stacking algorithm is employed to create an ensemble of strong learners. These learning algorithms can be finalized through experimentation. The model deployment is executed via the Flask platform and the model will be deployed via “pikl” files and on the IBM Cloud platform.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration and Login	Registration through Form Registration through Email Login through Email
FR-2	User should be able to input car details	Car information like date of purchase, price, damages incurred, etc are entered by the user
FR-3	User should be able to view past predictions	User can view the previous predictions the model has made on different cars and categorise according to the brand, type of car, date of purchase, etc.

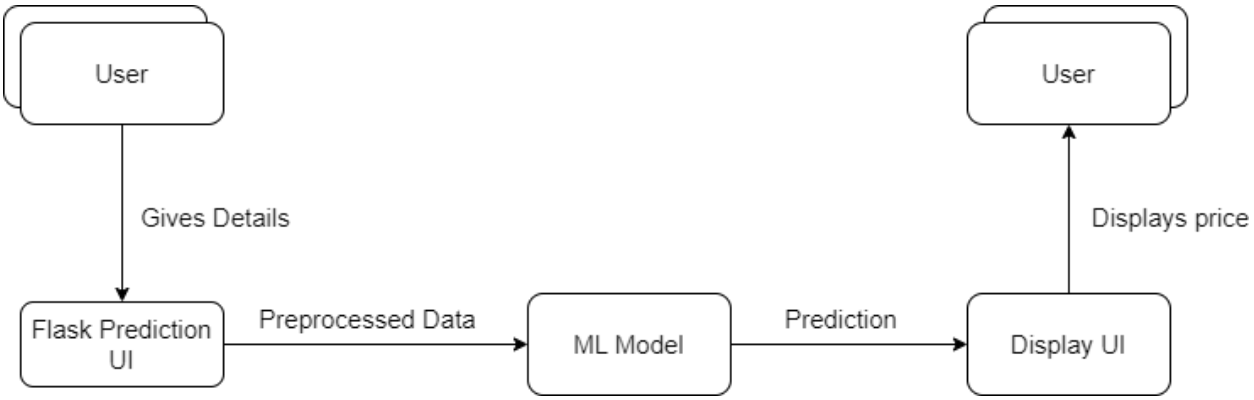
FR-4	System predicts car resale vale	Taking input features given by user, system should be able to predict car price by forwarding the prediction request to the ML model.
FR-5	Admin should be notified of any errors in the system	Any error that occurs like the model taking a long time to evaluate resale price should be notified to the admin so that the problem might be fixed.

4.2 Non-Functional requirements

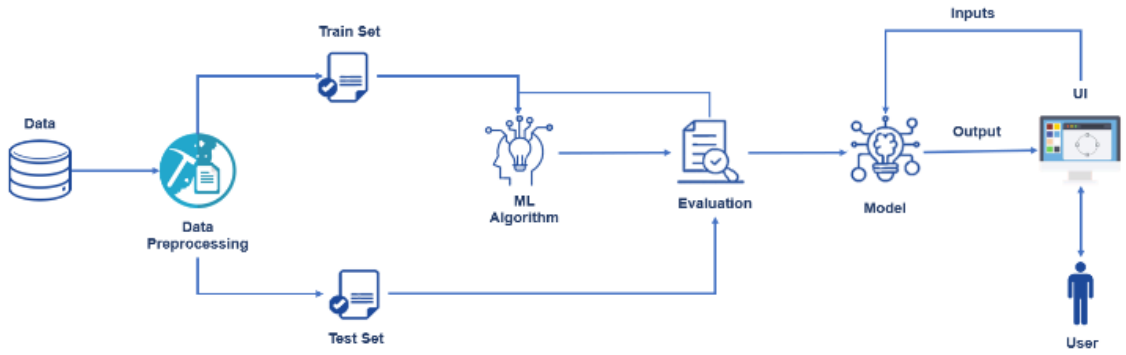
NFR-1	**Usability**	Effective User Interface with descriptions for each feature and proper layout that ensures each user finds it easy to access and interact with the system.
NFR-2	**Security**	Account creation for each user with a mandatory password strength check while creating the account.
NFR-3	**Reliability**	Chance of critical failure should be less than or equal to 2%.
NFR-4	**Performance**	The system must provide a webpage rendering images and texts upon receiving a request within a time of 8 seconds over a standard internet connection.
NFR-5	**Availability**	The website should be available to users 24x7. Any issues or errors will be addressed within the next 24 hours.
NFR-6	**Scalability**	The system must be scalable enough to support 1,00,000 requests at the same time without crashing.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirements	User Story Number	User Story	Acceptance Criteria	Priority	Release
Customer	Entering Car Details	USN-1	The user enters the necessary car details	All the mandatory fields are filled	High	Sprint 1
Customer	Viewing valuation	USN-2	The user's car's value is predicted by model	The value is displayed	High	Sprint 1
Admin	Updating model	USN-3	The admin can update the ML model after modifying it.	The ML model is properly loaded into the app.	High	Sprint 2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password
Sprint-2	Model Development		Gather dataset and perform pre-processing and cleaning
Sprint-2			Training and testing of model using data at hand

Sprint-2			Integrating model in application
Sprint-3	Input for prediction	USN-3	As a user, I should be able to give inputs to the application that will be sent to the model for prediction
Sprint-3	Prediction	USN-4	As a user, I will be able to view the predicted valuation for given vehicle
Sprint-4	Previous predictions	USN-5	As a user, I will be able to view previous predictions made by the model
Sprint-4	Update model	USN-6	As an admin, I will be able to update the model according to the fluctuating environment

6.2 Sprint Delivery Schedule

Sprint	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**
Sprint-1	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	6 Days	07 Nov 2022	12 Nov 2022
Sprint-4	6 Days	14 Nov 2022	19 Nov 2022

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Features added

7.1.1 Login,Register

Users can log into the application like a fully deployed website logging in or creating an account using the register feature.

7.1.2 Update

Users can update their password or account details.

7.2 Database Schema

Database made using inbuilt sqllite3 library available in python

User
id INTEGER PRIMARY KEY email TEXT NOT NULL username TEXT NOT NULL roll_number INTEGER NOT NULL pass_word TEXT NOT NULL

8. TESTING

8.1 Test Cases

TestcasesIBM.xlsx

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Car resale value prediction project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20

Duplicate	0	0	0	0	0
External	1	0	0	0	1
Fixed	5	0	0	20	25
Not Reproduced	0	0	0	0	0
Skipped	2	0	0	0	2
Won't Fix	0	0	0	0	0
Totals	18	4	2	23	51

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	5	0	0	5

Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics

RMSE: Root Mean Squared Error is the metric used by us to evaluate our model. It is most commonly used to evaluate regression models. Root Mean Square Error as the name suggests is calculated as the root of the mean squared errors of the predicted values. The formula is given below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Where zfi is the predicted value, zoi is the actual value and N is the sample size

We have used the sklearn.metrics package to calculate the RMSE.

Our ensembled model had a RMSE value of 3545.68 and the individual model error rates are given below:

Decision Tree: 4136.24

Random Forest: 3167.32

XGBoost: 3333.48

10. ADVANTAGES & DISADVANTAGES

Advantages:

- **The model is fairly accurate and is able to give a good prediction of what the actual resale value might be.**
- **The model is very quick in calculating the predictions.**
- **Errors of one model will be reduced by the ensembling with other models.**
- **It is easier for us to upload a better trained version of the model onto the cloud.**

Disadvantages:

- **The datasets available and the dataset the model has trained on do not give sufficient information to the model as it does not have very useful information.**
- **Attributes such as Fuel Mileage and Popularity of model has not been taken into account which can give a better idea about the resale value**

11. CONCLUSION

Thus with our project we can help both working resale dealers in the market and general users who want to sell their used cars by providing them with accurate resale value prediction.

12. FUTURE SCOPE

Can help with shaping the future of car resale market as our project opens up an insight into how the factors can be taken into for predicting the value of used cars.

13. APPENDIX

Source Code:

HTML FILES:

home.hmtl

```
<!DOCTYPE html>
<html lang="en">
  <style>
    a:link, a:visited {
      background-color: white;
      color: black;
      border: 2px solid black;
      border-radius: 25px;
      padding: 10px 20px;
```

```
        text-align: center;
        text-decoration: none;
        display: inline-block;
      }

      a:hover, a:active {
        background-color: beige;
        color: black;
        text-decoration: none;
      }
      </style>

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
                                                    <link                rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRx
T2MZw1T" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home</title>
</head>

<body>
  <div class="col-md-8">
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div class="alert alert-{{category}}">
            {{ message }}
          </div>
        {% endfor %}
```

```
        {% endif %}
      {% endwith %}
      {% block content %} {% endblock %}
  </div>
  <center>
  <h1>Car Resale Value Predictor</h1> <br>
  <a href="{{ url_for('login') }}">Click to Login</a><br> <br>
  <a href="{{ url_for('register') }}">Register here</a><br> <br>
  <a href="{{ url_for('update') }}">Update Password</a><br> <br>
  </center>

                                <script          src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi
6jizo" crossorigin="anonymous"></script>
            <script    src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86
dIHNDz0W1" crossorigin="anonymous"></script>
             <script    src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B0
7jRM" crossorigin="anonymous"></script>
</body>

</html>
```

Welcome.html(taking in inputs from the user)

```
<!DOCTYPE html>
<html lang="en">

<head>
                                                          <link                rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRx
T2MZw1T" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}" />
```

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome Page</title>
</head>

<body>
    <marquee>Welcome!</marquee> <br>
    <center>
    <h3>Enter Car details</h3>
    </center>

    <center>
    <form method="POST" action="/predict">
        <label for="sell">Choose a Seller:</label>
        <select id="sell" name="sell">
        <option value=0>Commercial</option>
        <option value=1>Private</option>
        </select>
        <br>
        <label for="ot">Choose a OfferType:</label>
        <select id="ot" name="ot">
        <option value=0>Offer</option>
        <option value=1>Request</option>
        </select>
        <br>
        <label for="vt">Choose a Vehicle type:</label>
        <select id="vt" name="vt">
        <option value=3>Coupe</option>
        <option value=1>Combination</option>
        <option value=0>Bus</option>
        <option value=2>Convertible</option>
        <option value=4>Limousine</option>
        <option value=7>Small</option>
```

<option value=8>SUV</option>
<option value=6>Others</option>
<option value=5>Not Declared</option>
</select>

<label for="gb">Choose a Gearbox:</label>
<select id="gb" name="gb">
<option value=0>Automatic</option>
<option value=1>Manual</option>
<option value=2>Not Declared</option>
</select>

PowerPs
<input name="pps" required>
kilometers
<input name="km" required>

<label for="ft">Choose a FuelType:</label>
<select id="ft" name="ft">
<option value=7>Petrol</option>
<option value=1>Diesel</option>
<option value=0>CNG</option>
<option value=4>LPG</option>
<option value=3>Hybrid</option>
<option value=2>Electric</option>
<option value=5>Not Declared</option>
<option values=6>Others</option>
</select>

<label for="brand">Choose a Brand:</label>
<select id="brand" name="brand">
<option value=0>Alpha Romeo</option>
<option value=1>Audi</option>
<option value=2>BMW</option>

<option value=3>Chevrolet</option>
<option value=4>Chrysler</option>
<option value=5>Citroen</option>
<option value=6>Dacia</option>
<option value=7>Daewoo</option>
<option value=8>Daihatsu</option>
<option value=9>Fiat</option>
<option value=10>Ford</option>
<option value=11>Honda</option>
<option value=12>Hyundai</option>
<option value=13>Jaguar</option>
<option value=14>Jeep</option>
<option value=15>Kia</option>
<option value=16>Lada</option>
<option value=17>Lancia</option>
<option value=18>Land Rover</option>
<option value=19>Mazda</option>
<option value=20>Mercedes</option>
<option value=21>Mini</option>
<option value=22>Mitsubishi</option>
<option value=23>Nissan</option>
<option value=24>Opel</option>
<option value=25>Peugeot</option>
<option value=26>Porsche</option>
<option value=27>Renault</option>
<option value=28>Rover</option>
<option value=29>Saab</option>
<option value=30>Seat</option>
<option value=31>Skoda</option>
<option value=34>Subaru</option>
<option value=35>Suzuki</option>
<option value=36>Toyota</option>
<option value=38>Volkswagen</option>
<option value=39>Volvo</option>

```
    </select>
    <br>
    <label for="nr">Repaired?</label>
    <select id="nr" name="nr">
    <option value=1>Yes</option>
    <option value=0>No</option>
    <option value=2>Not Declared</option>
    </select>
    <br>
    <p>Age(in months)</p>
    <input name="age" required>
    <br>
    <br>
    <button type="submit">Submit</button>
  </form>
  </center>

  <br>
  <br>
  <a href="{{url_for('logout')}}">Press here to logout</a>

                              <script          src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi
6jizo" crossorigin="anonymous"></script>
          <script    src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86
dIHNDz0W1" crossorigin="anonymous"></script>
           <script    src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B0
7jRM" crossorigin="anonymous"></script>
</body>

</html>
```

Predict.html(For displaying the output)

```
<!DOCTYPE html>
<html lang="en">
   <style>
      body{
         background-color: pink;
         color:black;
      }
   </style>
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Car Value</title>
</head>
<body>
   <h1>Car Value Predicted is</h1>

   <h1>{{predict1}}</h1>
   <br>
   <br>



</body>
</html>
```

app-ibm.py(FLASK FRAMEWORK)

```
from flask import Flask,request, render_template, url_for, redirect, flash
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from wtforms.validators import InputRequired, Length, ValidationError
from flask_bcrypt import Bcrypt
```

```
from wtforms import StringField, PasswordField, SubmitField, IntegerField
from flask_wtf import FlaskForm
import sqlite3
import joblib
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "iX5xF0JyPhYIfWGg37VPe14p1D7OMDPgPG1cl1yQJfJg"
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__)
bcrypt = Bcrypt(app)
app.config['SECRET_KEY'] = 'B7-1A3E'



login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'



@login_manager.user_loader
def load_user(user_id):
    conn = connect_db()
    user = conn.execute('SELECT * FROM user WHERE id = ?',
                        (user_id,)).fetchone()
    usr_obj = User(user[0], user[1], user[2])
    return usr_obj
```

```
def connect_db():
    conn = sqlite3.connect('database.db')
    return conn

class User:

    def __init__(self, id, email, username):
        self.id = id
        self.username = username
        self.email = email

    def to_json(self):
        return {"username": self.username,
                "email": self.email}

    def is_authenticated(self):
        return True

    def is_active(self):
        return True

    def is_anonymous(self):
        return False

    def get_id(self):
        return str(self.id)

class RegisterForm(FlaskForm):
    email = StringField(validators=[
        InputRequired(), Length(min=4, max=50)], render_kw={"placeholder": "Email"})
    username = StringField(validators=[
```

```
                        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder":
"Username"})
    rollnumber = StringField(validators=[
        InputRequired(), Length(min=5, max=10)], render_kw={"placeholder": "RollNumber"})
    password = PasswordField(validators=[
                        InputRequired(), Length(min=8, max=20)], render_kw={"placeholder":
"Password"})

    submit = SubmitField('Register')

    def validate_username(self, username):
        conn = connect_db()
        existing_user_username = conn.execute('SELECT * FROM user WHERE username = ?',
                                (username.data,)).fetchone()
        conn.commit()
        conn.close()
        if existing_user_username:
            raise ValidationError(
                'That username already exists. Try another one.')


class LoginForm(FlaskForm):
    username = StringField(validators=[
                        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder":
"Username"})

    password = PasswordField(validators=[
                        InputRequired(), Length(min=8, max=20)], render_kw={"placeholder":
"Password"})

    submit = SubmitField('Login')


class UpdateForm(FlaskForm):
```

```
    username = StringField(validators=[
                            InputRequired(), Length(min=4, max=20)], render_kw={"placeholder":
"Username"})

    oldpassword = PasswordField(validators=[
                InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Previous
Password"})

    password = PasswordField(validators=[
                            InputRequired(), Length(min=8, max=20)], render_kw={"placeholder":
"Password"})

    submit = SubmitField('Update')



@app.route('/')
def home():
    return render_template('home.html')



@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        conn = connect_db()
        user = conn.execute('SELECT * FROM user WHERE username = ?',
                            (form.username.data,)).fetchone()
        conn.commit()
        conn.close()
        if user:
            if bcrypt.check_password_hash(user[4], form.password.data):
                usr_obj = User(user[0], user[1], user[2])
                login_user(usr_obj)
                return redirect(url_for('welcome'))
```

```
        else:
            print('Hi')
            flash(f'Invalid credentials, check and try logging in again.', 'danger')
            return redirect(url_for('login'))

    return render_template('login.html', form=form)



@app.route('/welcome', methods=['GET', 'POST'])
@login_required
def welcome():
    return render_template('welcome.html')

@app.route('/predict', methods=['POST'])
def predictSpecies():
    sell = float(request.form['sell'])
    ot = float(request.form['ot'])
    vt = float(request.form['vt'])
    gb = float(request.form['gb'])
    pps=float(request.form['pps'])
    km=float(request.form['km'])
    ft=float(request.form['ft'])
    brand=float(request.form['brand'])
    nr=float(request.form['nr'])
    age=float(request.form['age'])
    arr = [[sell, ot, vt, gb,pps,km,ft,brand,nr,age]]


                payload_scoring = {"input_data": [{"field": [['sell', 'ot', 'vt',
'gb','pps','km','ft','brand','nr','age']], "values":arr}]}

                                                        response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/a4a92034-8fcd-4e79-ab7c-
```

```
521a5d8cb7d5/predictions?version=2022-11-15',                    json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
   print(response_scoring)
   predictions =  response_scoring.json()
   pr = predictions['predictions'][0]['values'][0][0]
   print("final prediction",pr)
   return render_template('predict.html',predict1=pr)



@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
   logout_user()
   return redirect(url_for('login'))


@ app.route('/register', methods=['GET', 'POST'])
def register():
   form = RegisterForm()
   conn = connect_db()
   if form.validate_on_submit():
      hashed_password = bcrypt.generate_password_hash(form.password.data)
      conn.execute('INSERT INTO user (email, username, roll_number, pass_word) VALUES (?,
?, ?, ?)',
                   (form.email.data, form.username.data, form.rollnumber.data, hashed_password))
      conn.commit()
      conn.close()
      return redirect(url_for('login'))
   return render_template('register.html', form=form)


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

```
    form = UpdateForm()
    conn = connect_db()
    if form.validate_on_submit():
        conn = connect_db()
        user = conn.execute('SELECT * FROM user WHERE username = ?',
                            (form.username.data,)).fetchone()
        if user:
            if bcrypt.check_password_hash(user[4], form.oldpassword.data):
                print(user)
                hashed_password1 = bcrypt.generate_password_hash(
                    form.password.data)
                conn.execute('UPDATE user set pass_word = ? where username = ?',
                             (hashed_password1, form.username.data))
                conn.commit()
                conn.close()
                flash(f'Password changed successfully.', 'success')
                return redirect(url_for('home'))
            else:
                flash(f'Invalid password, Enter valid password.', 'danger')
                return redirect(url_for('update'))
        else:
            flash(f'Invalid user, Enter valid User.', 'danger')
            return redirect(url_for('update'))
    return render_template('update.html', form=form)


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

GitHub & Project Demo Link

GitHub Repo Link:https://github.com/IBM-EPBL/IBM-Project-13666-1659525228

Project Demo:Click Here