**CERTIFICATE**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

This is to certify that the mini project entitled **“USED CARS PRICE PREDICTION”** that is being submitted by **MATSA MURALI KRISHNA** with Reg. No. **322203320038**, student of **Gayatri Vidya Parishad College of Engineering (Autonomous),** during the academic year 2023 - 2024.

### Internal Guide Head of the Department

### Dr N.Jaya Lakshmi Dr Y.Anuradha M.Tech,PhD

### Associate Professor Associate Professor and Head

### Department of Computer Application Department of Computer Application

**MINI PROJECT**

#### USED CARS PRICE PREDICTION

#### INDUSTRY ORIENTED MINI PROJECT

#### Master of Computer Applications (MCA)

*Submitted By*

### MATSA MURALI KRISHNA

### Reg no: 322203320038

### *Under the esteemed guidance of*

### Dr N.Jaya Lakshmi

### Associate Professor

### Department of Computer Applications

### GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)

### (Affiliated to Andhra University, Vishakhapatnam, A.P)

**TABLE OF CONTENTS**

**ABSTRACT 5**

**PROBLEM STATEMENT 6**

**EXITING SYSTEM 7**

**PROPOSED SYSTEM 8**

**SYSTEM REQUIREMENTS 9**

**TOOLS AND PLATFORM 10**

**ARCHITECTURE 11**

**ALGORITHM 12**

**SAMPLE CODE 13-21**

**WEBPAGE PREVIEW 22**

**FINAL OUTPUT 23**

**CONCLUSION 24**

# ABSTRACT

Everyone has a dream to buy a new car. But they cannot afford to buy a new car at that time he wants to buy a used cars. They do not have a knowledge to predict the price of a car. At that time this project will be helpful to predict the car selling price based on their features like the car model, the number of years that a car is old, the type of fuel it uses, the type of seller, the type of transmission and the number of kilometers that the car has driven so far. This project will help to get an approximation about selling price of a used car based on its features and reduces the seller and consumer risk in business. The proposed model utilizes the machine learning algorithms and regression techniques of statistics decision tree, XG boost and random forest regressions to achieve this task.

**PROBLEM-STATEMENT:**

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

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

• Model of the car namely Ford Figo, Hyundai Creta

• Year of manufacturing like 2020, 2021,2022

• Type of fuel namely Petrol, Diesel

• Price range or Budget

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

• Mileage

**EXISTING SYSTEM**

Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and it’s 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.

Disadvantages:

1. Not accurate
2. Not effective

**PROPOSED SYSTEM**

To overcome this problem we have developed a model which will be highly effective. Machine learning Algorithm is used because they provide continuous value as an output and not a categorized 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

**SYSTEM REQUIREMENTS**

* OS : Windows 10 or 11
* RAM : 4GB(minimum required)
* IDE : Visual Studio
* Framework : FLASK
* Script Language :Python
* Frontend :HTML,CSS
* Processor :i3(minimum required)
* Hard Disk :128 GB

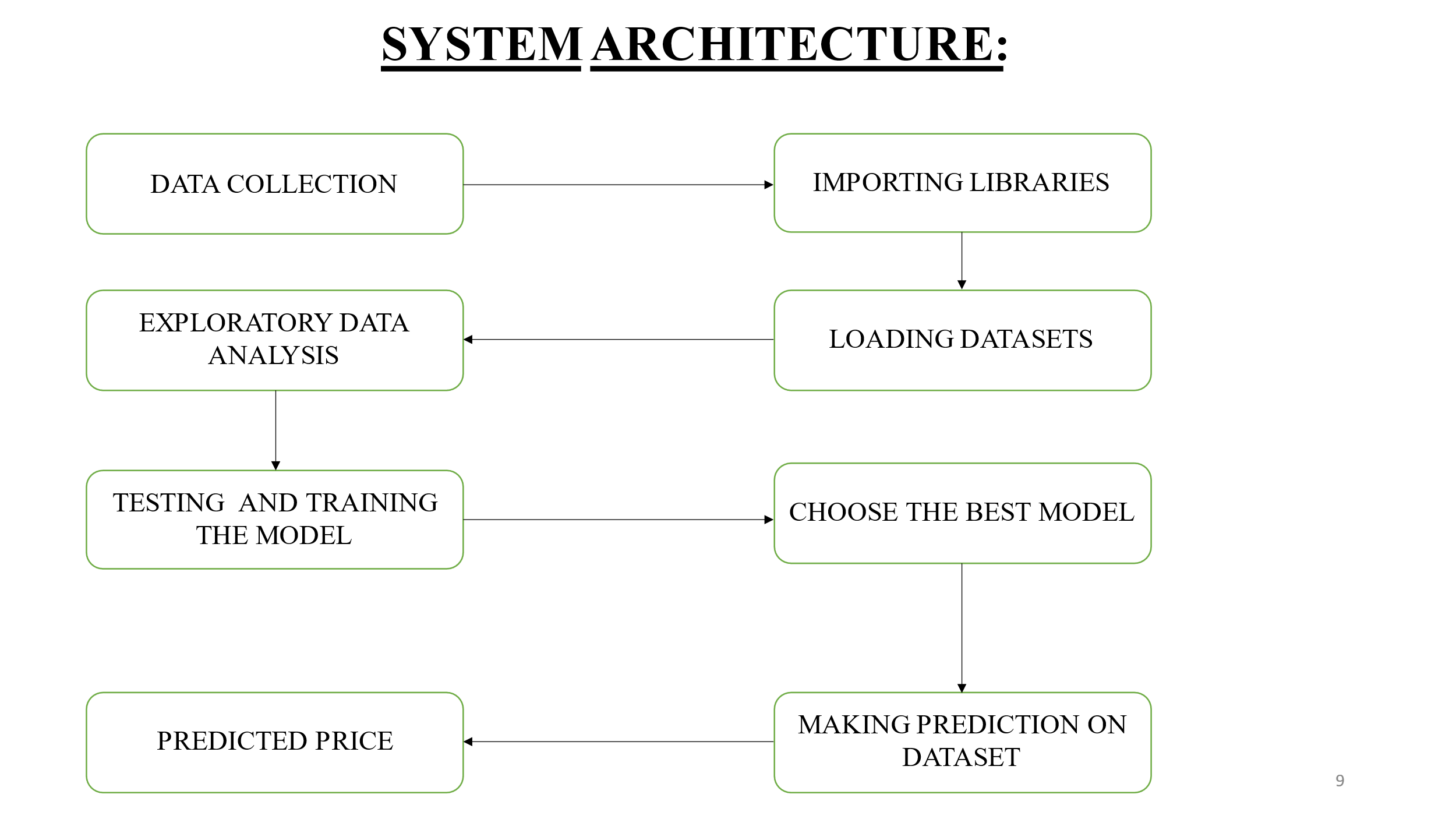
**TOOLS AND PLATFORM**

**HTML**: HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g.: HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

**CSS:** Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a web page should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

**Flask:** Flask is a lightweight and versatile web framework for Python, designed to simplify the development of web applications. Known for its simplicity and modularity, Flask provides essential tools and libraries, allowing developers to build scalable and efficient web solutions. With its minimalist design, Flask empowers developers to choose and integrate components as needed, making it an ideal choice for both beginners and experienced developers seeking flexibility and ease of use in web application development.

**Python**: Python is a versatile and powerful programming language that is widely used in various fields, including data science, machine learning, web development, and automation. The simplicity of Python allows for faster development and easier maintenance of code. Python is open source, allowing developers to contribute to the language's development and benefit from a wealth of open-source libraries and tools. NumPy and Pandas provides Essential for data manipulation and analysis. Scikit-learn Provides a simple and efficient tool for data mining and data ana



**ALGORITHM:**

XGBoost is a robust machine-learning algorithm that can help you understand your data and make better decisions. XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models. XGBoost is designed for speed, ease of use, and performance on large datasets.

Initialization:

* 1. Set an initial prediction (usually the mean of the target variable).
  2. Calculate the initial residuals.

For Each Iteration:

* 1. Train a weak learner (decision tree) on the negative gradients of the loss function.
  2. Calculate the learning rate-adjusted contribution of the tree to the model.
  3. Update predictions and residuals.

Regularization:

* 1. Apply L1 and L2 regularization to control the complexity of the model.

Stopping Criteria:

* 1. Determine when to stop adding trees based on a user-defined number of iterations or when no significant improvement is observed.

**SAMPLE CODE:**

**STYLE.CSS:**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

background-image: url('download.jpeg');

}

.container {

max-width: 600px;

margin: 15px auto;

background-color: #fff;

padding: 5px;

border-radius: 6px;

box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);

}

form {

display: flex;

flex-direction: column;

}

label {

margin-bottom: 5px;

}

input,

select {

margin-bottom: 5px;

padding: 3px;

}

button {

background-color: #4caf50;

color: #fff;

padding: 10px;

border: none;

cursor: pointer;

}

Button :hover {

background-color**: #**45a049;

}

.result {

margin-top: 10px;

font-weight: bold;

}

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<title>Car Price Prediction</title>

</head>

<body>

<div class="container">

<h1>Car Price Prediction</h1>

<form action="{{ url\_for('predict') }}" method="post">

<label for="Car\_Name">Car\_Name:</label>

<input type="text" name="Car\_Name" required><br>

<label for="Present\_Price">Present\_Price:</label>

<input type="text" name="Present\_Price" placeholder="Enter the amount in lakhs"required><br>

<label for="Kms\_Driven">Kms\_Driven:</label>

<input type="text" name="Kms\_Driven"placeholder="Enter the Number of kilometers driven by the car" required><br>

<label for="Fuel\_Type">Fuel\_Type:</label>

<select name="Fuel\_Type" required>

<option value="select">Select</option>

<option value="0">Petrol</option>

<option value="1">Diesel</option>

<option value="2">CNG</option>

</select><br>

<label for="Seller\_Type">Seller\_Type:</label>

<select name="Seller\_Type" required>

<option value="select">Select</option>

<option value="0">Dealer</option>

<option value="1">Individual</option>

</select><br>

<label for="Transmission">Transmission:</label>

<select name="Transmission" required>

<option value="select">Select</option>

<option value="0">Manual</option>

<option value="1">Automatic</option>

</select><br>

<label for="Owner">Owner:</label>

<input type="text" name="Owner" required><br>

<label for="Age">Age:</label>

<input type="text" name="Age" required><br>

<button type="submit">Predict</button>

</form>

<div class="result" id="result">{{ result }}</div>

</div>

</body>

</html>

**app.py:**

from flask import Flask, render\_template, request

import joblib

import pandas as pd

from sklearn import preprocessing

app = Flask(\_\_name\_\_)

label\_encoder = preprocessing.LabelEncoder()

@app.route('/')

def index():

return render\_template('index.html', result="")

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

def predict():

try:

car\_name\_str = request.form['Car\_Name']

present\_price = float(request.form['Present\_Price'])

kms\_driven = float(request.form['Kms\_Driven'])

fuel\_type = int(request.form['Fuel\_Type'])

seller\_type = int(request.form['Seller\_Type'])

transmission = int(request.form['Transmission'])

owner = float(request.form['Owner'])

age = float(request.form['Age'])

car\_name\_numeric = label\_encoder.fit\_transform([car\_name\_str])

print(car\_name\_numeric)

data\_new = pd.DataFrame({

'Car\_Name': car\_name\_numeric,

'Present\_Price': present\_price,

'Kms\_Driven': kms\_driven,

'Fuel\_Type': fuel\_type,

'Seller\_Type': seller\_type,

'Transmission': transmission,

'Owner': owner,

'Age': age

}, index=[0])

model = joblib.load('car\_price\_predictor')

result = model.predict(data\_new)

return render\_template('index.html', result=f"Car Purchase amount: {result[0]}")

except Exception as e:

return render\_template('index.html', result=f"Error: {str(e)}")

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

app.run(debug=True)

:

**car.ipnyb:**

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

data = pd.read\_csv('carproject.csv')

from sklearn import preprocessing

#make an instance of Label Encoder

label\_encoder = preprocessing.LabelEncoder()

#a=preprocessing.LabelEncoder()

data['Car\_Name'] = label\_encoder.fit\_transform(data['Car\_Name'])

data

### 1. Display Top 5 Rows of The Dataset

#data=data.drop(columns=['Unnamed: 0','carid'])

data.head()

### 2. Check Last 5 Rows of The Dataset

data.tail()

### 3. Find Shape of Our Dataset (Number of Rows And Number of Columns)

data.shape

print("Number of Rows",data.shape[0])

print("Number of Columns",data.shape[1])

### 4. Get Information About Our Dataset Like the Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

data.info()

### 5. Check Null Values In The Dataset

data.isnull().sum()

### 6. Get Overall Statistics About The Dataset

data.describe()

### 7. Data Preprocessing

data.head(1)

import datetime

date\_time = datetime.datetime.now()

data['Age']=date\_time.year - data['Year']

data.head()

data.drop('Year',axis=1,inplace=True)

data.head()

#### Outlier Removal

import seaborn as sns

sns.boxplot(data['Selling\_Price'])

sorted(data['Selling\_Price'],reverse=True)

data = data[~(data['Selling\_Price']>=33.0) & (data['Selling\_Price']<=35.0)]

data.shape

#### Encoding the Categorical Columns

data.head(1)

data['Fuel\_Type'].unique()

data['Fuel\_Type'] = data['Fuel\_Type'].map({'Petrol':0,'Diesel':1,'CNG':2})

data['Fuel\_Type'].unique()

data['Seller\_Type'].unique()

data['Seller\_Type'] = data['Seller\_Type'].map({'Dealer':0,'Individual':1})

data['Seller\_Type'].unique()

data['Transmission'].unique()

data['Transmission'] =data['Transmission'].map({'Manual':0,'Automatic':1})

data['Transmission'].unique()

data.head()

### 8. Store Feature Matrix In X and Response(Target) In Vector y

X = data.drop(['Selling\_Price'],axis=1)

y = data['Selling\_Price']

y

### 9. Splitting The Dataset Into The Training Set And Test Set

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=42)

### 10. Import The models

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import GradientBoostingRegressor

from xgboost import XGBRegressor

### 11. Model Training

lr = LinearRegression()

lr.fit(X\_train,y\_train)

rf = RandomForestRegressor()

rf.fit(X\_train,y\_train)

xgb = GradientBoostingRegressor()

xgb.fit(X\_train,y\_train)

xg = XGBRegressor()

xg.fit(X\_train,y\_train)

### 12. Prediction on Test Data

y\_pred1 = lr.predict(X\_test)

y\_pred2 = rf.predict(X\_test)

y\_pred3 = xgb.predict(X\_test)

y\_pred4 = xg.predict(X\_test)

### 13. Evaluating the Algorithm

from sklearn import metrics

score1 = metrics.r2\_score(y\_test,y\_pred1)

score2 = metrics.r2\_score(y\_test,y\_pred2)

score3 = metrics.r2\_score(y\_test,y\_pred3)

score4 = metrics.r2\_score(y\_test,y\_pred4)

print(score1,score2,score3,score4)

final\_data = pd.DataFrame({'Models':['LR','RF','GBR','XG'],

"R2\_SCORE":[score1,score2,score3,score4]})

final\_data

#sns.barplot(final\_data['Models'],final\_data['R2\_SCORE'])

### 14. Save The Model

xg = XGBRegressor()

# print(X)

#rf = RandomForestRegressor()

xg\_final = xg.fit(X,y)

#rf\_final=rf.fit(X,y)

import joblib

joblib.dump(xg\_final,'car\_price\_predictor')

model = joblib.load('car\_price\_predictor')

### 15. Prediction on New Data

import pandas as pd

data\_new = pd.DataFrame({

'Car\_Name':90,

'Present\_Price':5.59,

'Kms\_Driven':27000,

'Fuel\_Type':0,

'Seller\_Type':0,

'Transmission':0,

'Owner':0,

'Age':8

},index=[0])

model.predict(data\_new)

### GUI

from tkinter import \*

import joblib

def show\_entry\_fields():

p=float(e.get())

p1=float(e1.get())

p2=float(e2.get())

p3=float(e3.get())

p4=float(e4.get())

p5=float(e5.get())

p6=float(e6.get())

p7=float(e7.get())

model = joblib.load('car\_price\_predictor')

data\_new = pd.DataFrame({

'Car\_Name':p,

'Present\_Price':p1,

'Kms\_Driven':p2,

'Fuel\_Type':p3,

'Seller\_Type':p4,

'Transmission':p5,

'Owner':p6,

'Age':p7

},index=[0])

result=model.predict(data\_new)

Label(master, text="Car Purchase amount").grid(row=9)

Label(master, text=result).grid(row=11)

print("Car Purchase amount", result[0])

master = Tk()

master.title("Car Price Prediction Using Machine Learning")

label = Label(master, text = "Car Price Prediction Using Machine Learning"

, bg = "black", fg = "white"). \

grid(row=0,columnspan=2)

Label(master,text="Car\_Name").grid(row=1)

Label(master, text="Present\_Price").grid(row=2)

Label(master, text="Kms\_Driven").grid(row=3)

Label(master, text="Fuel\_Type").grid(row=4)

Label(master, text="Seller\_Type").grid(row=5)

Label(master, text="Transmission").grid(row=6)

Label(master, text="Owner").grid(row=7)

Label(master, text="Age").grid(row=8)

e=Entry(master)

e1 = Entry(master)

e2 = Entry(master)

e3 = Entry(master)

e4 = Entry(master)

e5 = Entry(master)

e6 = Entry(master)

e7 = Entry(master)

e.grid(row=1,column=1)

e1.grid(row=2, column=1)

e2.grid(row=3, column=1)

e3.grid(row=4, column=1)

e4.grid(row=5, column=1)

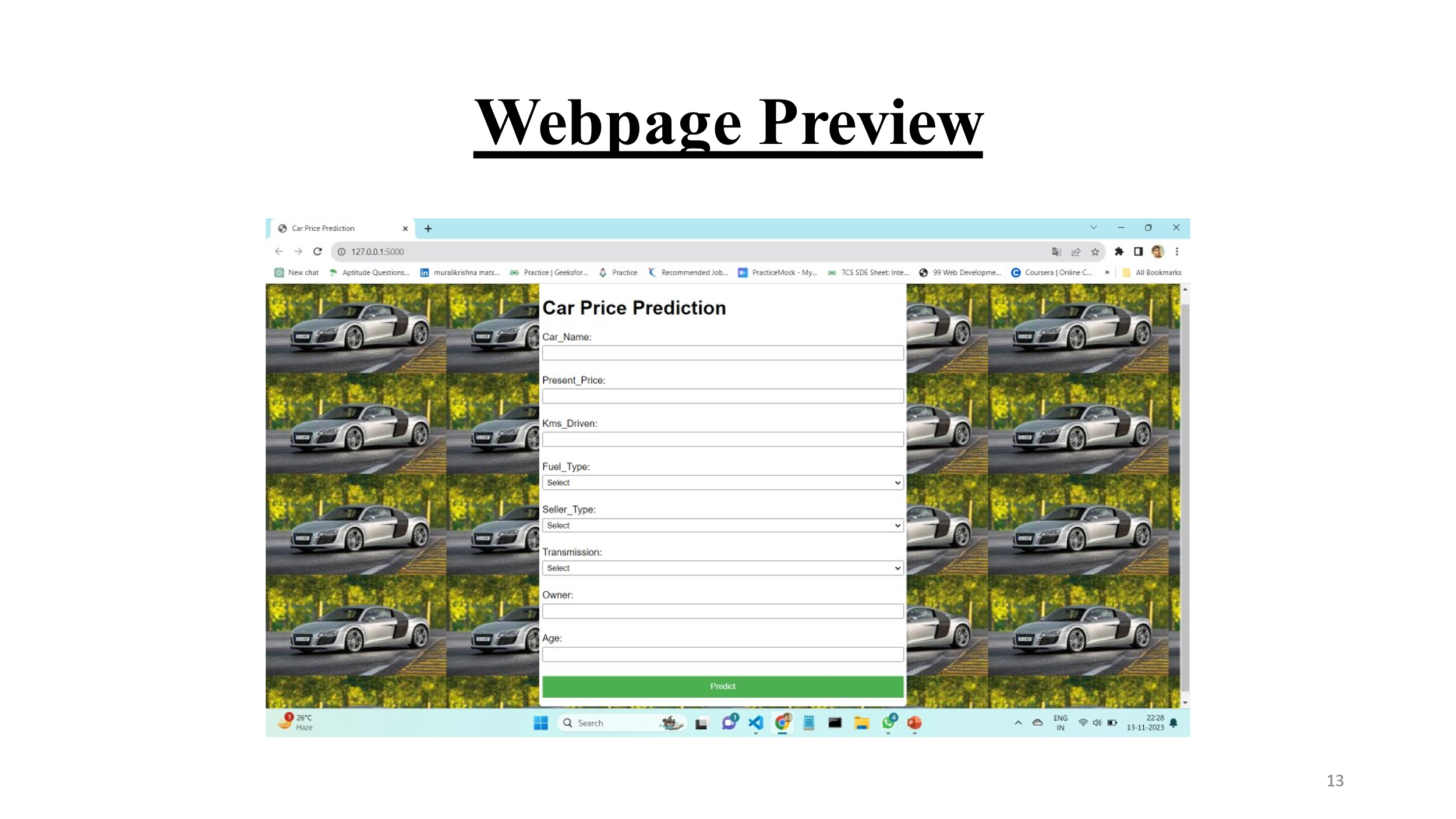
e5.grid(row=6, column=1)

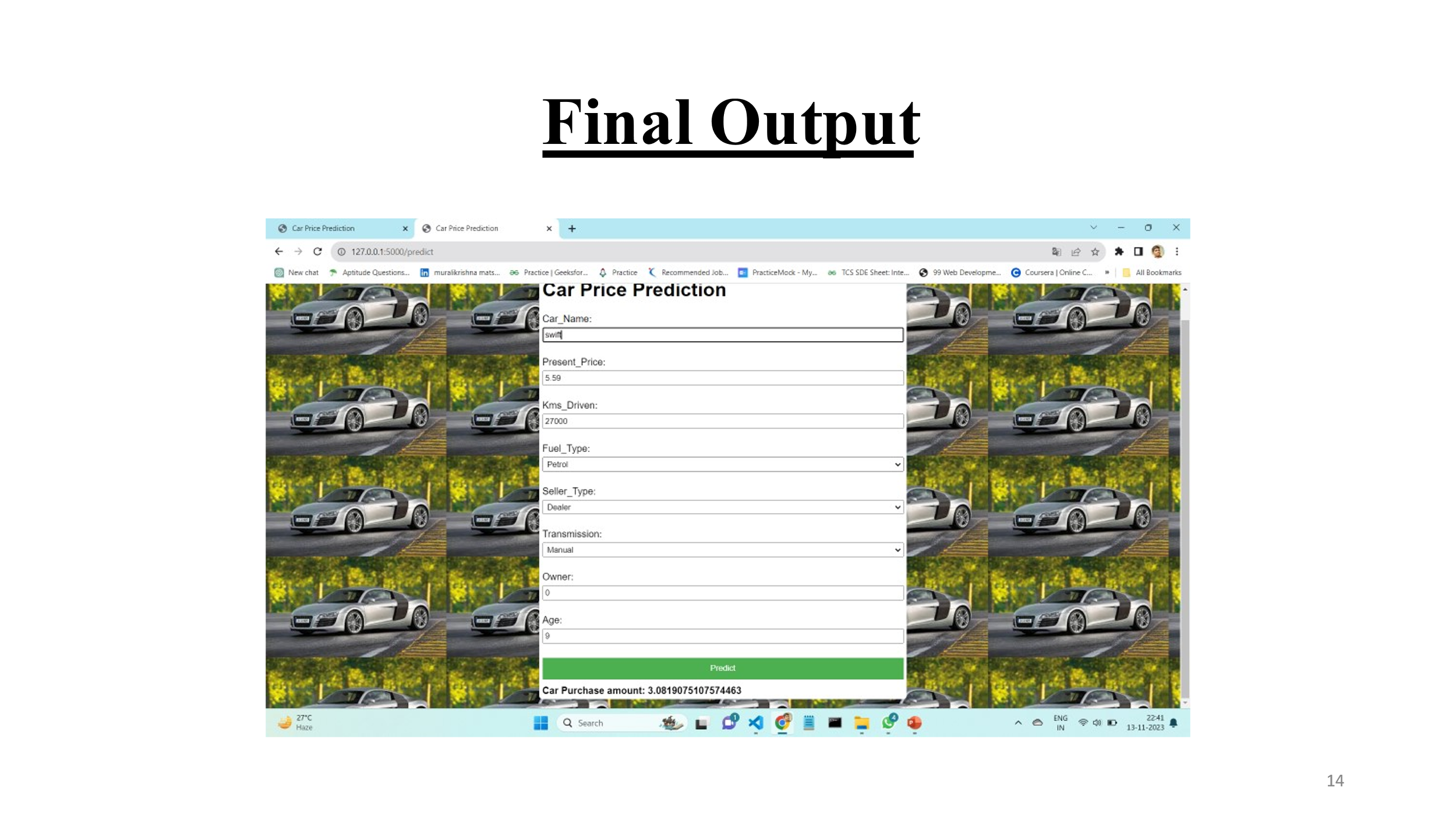
e6.grid(row=7, column=1)

e7.grid(row=8, column=1)

Button(master, text='Predict', command=show\_entry\_fields).grid()

mainloop()





**CONCLUSION**

* Predicting used car prices has practical implications for both buyers and sellers in the automotive market.
* Our model serves as a valuable decision support tool for users navigating the complex landscape of used car transactions.
* The successful implementation of our used car price prediction model has the potential to positively impact the automotive market, aiding both buyers and sellers in making informed decisions**.**