A

PROJECT REPORT

ON

Used Car Price Predictor

Submitted in partial fulfilment of the requirements of the degree

Bachelor of Engineering In Information Technology

By

Aditya Nar 34
Dattatray Narhe 35
Harsh Gawas 38
Harsh Pandey 10

Supervisor: Prof. Amarja Adgaonkar

Department of Information Technology

K.C. College of Engineering and Management Studies And Research,
Thane (E)

University of Mumbai

2023-24

CERTIFICATE

This is to certify that the project entitled **"Car Price Predictor"** is a bonafide work of **Aditya Nar (34) , Dattatray Narhe (35) , Harsh Pandey (38) , Harsh Gawas (10)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Information Technology"**.

Name and sign

Co Supervisor/Guide

Supervisor/Guide

Head of Department
Prof.Amarja Adgaonkar

Principal
Dr.Vilas Nitnaware

Project Report Approval for T.E

This project report entitled ***Car Price Predictor*** by ***Aditya Nar, Dattatray Narhe, Harsh Pandey, Harsh Gawas*** is approved for the degree of Bachelor of Engineering in **Information Technology**

Examiners

1.----------

2.----------

Date:

Place:

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

_____

(Aditya Nar - 34)

(Dattatray Narhe-35)

(Harsh Pandey – 38)

(Harsh Gawas - 10)

Date:

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

_____

(Aditya Nar - 34)

(Dattatray Narhe-35)

(Harsh Pandey – 38)

(Harsh Gawas - 10)

Date:

ACKNOWLEDGEMENT

We would like to express special thanks of gratitude to our guide **Mrs.Punam Bagul** as well as our Project Coordinator who gave us the golden opportunity to do this wonderful project on the topic of **Car Price Predictor** which also helped us in doing a lot of research and we came to know about so many new things. We are very grateful to our Head of the Department **Mrs.Amarja Adgaonkar** for extending her help directly and indirectly through various channels in our project work**.** We would also like to thank Principal **Dr. Vilas Nitnaware** for providing us the opportunity to implement our project. We are really thankful to them. Finally we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Thanking You.

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

The Used Car Price Predictor project leverages machine learning (ML) techniques to forecast the prices of pre-owned vehicles accurately. This project addresses the challenge faced by both buyers and sellers in determining fair market values for used cars, considering numerous variables that influence pricing. By employing advanced ML algorithms, such as regression models and ensemble methods, this system analyzes historical data encompassing car specifications, mileage, age, brand, location, and market trends. Feature engineering techniques are applied to extract meaningful insights, while model evaluation and hyperparameter tuning ensure robust predictions. The project's outcome is a user-friendly interface where users input car details, and the system provides an estimated price range based on the trained ML models. The Used Car Price Predictor not only enhances transparency and efficiency in the used car market but also serves as a valuable tool for informed decision-making by both buyers and sellers.

Furthermore, the project encompasses a user-friendly web interface or mobile application where users can input specific car details and receive a predicted price range based on the trained models. This predictive tool not only benefits individual buyers and sellers but also contributes to market transparency and fairness by establishing a more standardized pricing mechanism for used cars. Additionally, the project could be expanded to include real-time data integration, sentiment analysis of customer reviews, and incorporation of additional features for even more precise price predictions.

1. Introduction

The used car market is a dynamic and complex ecosystem, influenced by a myriad of factors such as vehicle age, mileage, brand reputation, regional demand, and market trends. However, pricing these pre-owned vehicles accurately remains a significant challenge for both buyers and sellers. Buyers often struggle to gauge fair market values, leading to uncertainty and potential overpayment. On the other hand, sellers may face challenges in setting competitive prices that attract buyers without undervaluing their vehicles.

The advent of machine learning (ML) technologies has revolutionized various industries, including automotive, by offering data-driven insights and predictive capabilities. The Used Car Price Predictor project harnesses the power of ML algorithms to address the pricing dilemma in the used car market. By leveraging historical data encompassing a wide range of vehicle attributes and market dynamics, this project aims to develop a robust predictive model that can accurately estimate the prices of used cars.

The significance of this project lies in its potential to enhance transparency, efficiency, and fairness within the used car market. A reliable price prediction system not only benefits individual buyers and sellers but also contributes to overall market equilibrium by aligning prices more closely with actual value propositions. Furthermore, the project sets the stage for future advancements in automotive pricing analytics, with opportunities to integrate real-time data streams, sentiment analysis, and additional features for more nuanced predictions.

In this paper, we delve into the methodology, data processing techniques, ML algorithms employed, model evaluation metrics, and the envisioned user interface of the Used Car Price Predictor system. Through this exploration, we aim to demonstrate the value and feasibility of leveraging ML in solving real-world challenges within the automotive industry.

2. Literature Survey

Year	Title	Methodology	Advantages	Limitations
2021	"Predicting Used Car Prices Using Machine Learning"	- Data collection from online listings - Preprocessing (imputation, encoding) - Regression models	- Accurate price predictions based on historical data - Scalable for large datasets - Simple model interpretation	- Limited to regression models, may not capture complex patterns - Relies heavily on data quality and availability
2020	"Ensemble Learning for Used Car Price Prediction"	- Ensemble methods (random forests, gradient boosting) - Feature engineering	- Improved predictive accuracy through ensemble techniques - Handles nonlinear relationships in data	- Increased computational complexity compared to individual models - Requires careful tuning of hyperparameters for optimal performance
2019	"Deep Learning Approaches for Predicting Used Car Prices"	- Deep learning models (neural networks, LSTM) - Feature scaling and normalization	- Captures complex patterns and nonlinear relationships in data - Potential for automatic feature extraction	- Requires large amounts of data for training deep learning models effectively - Prone to overfitting if not properly regularized and validated
2018	"Hybrid Approach for Used Car Price Prediction"	- Hybrid of regression and clustering techniques - Feature selection and extraction	- Integrates multiple modeling techniques for robust predictions - Addresses heterogeneity in data	- Complexity in combining different models and methodologies - Interpretability may be compromised with hybrid approaches
2017	"Sentiment Analysis in Used Car Price Prediction"	- Sentiment analysis of customer reviews - Feature engineering for sentiment features	- Incorporates qualitative data (customer sentiments) into pricing predictions - Provides additional context for pricing decisions	- Relies on accurate sentiment analysis tools and datasets - Limited to availability of customer reviews and sentiment data

3.1 Proposed Work

3.1 Requirement Analysis

3.1.1 Scope

1. Data Scope:
- The project will primarily focus on utilizing historical data related to used cars, including attributes such as make, model, year, mileage, condition, region, and past sales prices.
- Data collection may involve scraping online listings, leveraging existing datasets, or collaborating with automotive industry partners for access to proprietary data.

2. Geographical Scope:
- The initial scope of the project will cover a specific geographical region or market segment, such as a particular country or region with well-defined automotive market characteristics.
- Expansion to include global market data or multiple regional markets can be considered in future iterations for broader applicability.

3. Predictive Modeling Scope:
- The project aims to develop and deploy machine learning models for predicting price ranges of used cars based on input parameters provided by users.
- ML algorithms such as decision trees, random forests, gradient boosting, or regression models will be explored for their suitability and predictive accuracy.

4. User Interface Scope:
- A user-friendly web interface or mobile application will be developed to facilitate user interactions with the Used Car Price Predictor system.
- The interface will allow users to input car details and receive estimated price ranges, along with potential visualizations or insights to aid decision-making.

5. Enhancement Opportunities:
- The project leaves room for enhancements such as real-time data integration, sentiment analysis of customer reviews, and incorporation of additional features (e.g., vehicle specifications, market trends) for more nuanced predictions.
- Integration with external APIs or data sources may be explored to enrich the predictive capabilities of the system.

6. Performance Metrics:
- Model performance will be evaluated using metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared (R2) to ensure predictive accuracy and reliability.
- Scalability and performance optimization strategies will be considered to handle increased user traffic and data volume.

7. Documentation and Deployment:
- Comprehensive documentation will be created covering system architecture, data sources, ML algorithms, user guidelines, and maintenance procedures.
- The system will be deployed on a suitable hosting platform, ensuring scalability, security, and reliability for end-users.

3.1.2 Feasibility Study

A feasibility study assesses the viability and practicality of a project, considering various factors such as technical, economic, operational, and scheduling aspects. Let's break down the feasibility study for the Used Car Price Predictor project:

1. Technical Feasibility:
- Data Availability:Assess the availability and accessibility of comprehensive used car data, including attributes like make, model, year, mileage, condition, region, and historical sales prices. Evaluate data sources for reliability and relevance.
- ML Model Development: Determine the feasibility of implementing ML algorithms (e.g., decision trees, random forests, regression models) for price prediction based on available data. Consider the complexity of feature engineering, model training, and optimization.
- Scalability:Evaluate the scalability of the system in handling large datasets, concurrent user requests, and potential future enhancements such as real-time data integration and additional features.

2. Economic Feasibility:
-Cost Analysis: Conduct a cost-benefit analysis to determine the financial feasibility of the project. Evaluate costs related to data acquisition, software development, hardware infrastructure, maintenance, and ongoing support.
- Return on Investment (ROI): Estimate the potential ROI based on projected benefits such as increased market competitiveness, reduced pricing disparities, improved user experience, and potential revenue streams (e.g., subscription models, partnerships with automotive industry stakeholders).

3. Operational Feasibility:
- System Integration: Assess the feasibility of integrating the Used Car Price Predictor system with existing platforms, databases, and APIs for seamless data exchange and functionality.
- User Adoption: Evaluate user acceptance and adoption of the system by buyers, sellers, and other stakeholders in the used car market. Consider user feedback, usability testing, and training requirements.

4. Scheduling Feasibility:
- Timeline:Develop a realistic project timeline considering phases such as data collection, preprocessing, ML model development, user interface design, testing, deployment, and ongoing maintenance.
- Resource Allocation:Evaluate resource availability (e.g., development team, computing resources, budget) and allocate tasks effectively to meet project milestones within the specified timeframe.

3.1.3 Hardware and Software Requirement

Hardware Requirements:

1. Server Infrastructure:
- High-performance server or cloud-based computing resources for model training and inference.
- Adequate storage capacity to handle large datasets and model artifacts.
- Recommended specifications: multicore CPU (e.g., Intel Core i7 or AMD Ryzen 7), sufficient RAM (at least 16GB), and SSD storage for faster data access.

2. Networking Equipment:
- Reliable internet connectivity with sufficient bandwidth to support concurrent user requests.
- Networking hardware such as routers, switches, and firewalls for secure data transmission.

3. End-user Devices:
- Users can access the system via desktop computers, laptops, tablets, or smartphones.
- Ensure compatibility with various operating systems (Windows, macOS, Linux) and web browsers (Chrome, Firefox, Safari, Edge).

Software Requirements:

1. Operating System:
- Server-side: Linux distributions (e.g., Ubuntu Server, CentOS) or Windows Server for hosting the application and ML models.
- Development environments: Windows, macOS, or Linux for software development and testing.

2. Programming Languages and Libraries:
- Python for ML model development and backend logic.
- Frameworks/libraries such as TensorFlow, Scikit-learn, Pandas, NumPy for ML algorithms, data manipulation, and analysis.

3. Web Development Tools:
- Frontend: HTML5, CSS3, JavaScript (React, Angular, Vue.js) for building the user interface.
- Backend: Python web frameworks (Django, Flask) for server-side logic and API development.

4. Database Management:
- Relational database management system (RDBMS) such as PostgreSQL, MySQL, or SQLite for storing structured data.
- Optionally, NoSQL databases (MongoDB, Redis) for caching or storing unstructured data.

5. Development and Deployment Tools:
- Integrated Development Environments (IDEs): Visual Studio Code, PyCharm, Jupyter Notebooks for coding and testing ML models.
- Version control systems (e.g., Git) for collaboration and code management.

3.2 Problem Statement

The used car market presents a challenge for both buyers and sellers due to the lack of transparent and standardized pricing mechanisms. Buyers often struggle to determine fair market values for pre-owned vehicles, leading to uncertainty and potential overpayment. On the other hand, sellers face challenges in setting competitive prices that attract buyers without undervaluing their vehicles. This pricing disparity can result in inefficiencies, longer sales cycles, and dissatisfaction among market participants.

Additionally, the used car market is influenced by numerous factors such as vehicle age, mileage, brand reputation, regional demand, and market trends. The complex interplay of these variables makes manual pricing assessments time-consuming and prone to errors. As a result, there is a growing demand for data-driven solutions that leverage machine learning (ML) techniques to provide accurate and unbiased price predictions for used cars.

The aim of the project is to develop a robust Used Car Price Predictor system that addresses these challenges by leveraging historical data, ML algorithms, and user-friendly interfaces. The system will empower buyers with accurate price estimates based on car specifications and market trends, enabling informed decision-making and fair transactions. Similarly, sellers will benefit from guidance on pricing their vehicles competitively, leading to reduced listing times and improved market competitiveness.

Overall, the project seeks to bridge the pricing gap in the used car market, enhance market transparency, and streamline the buying and selling process for all stakeholders involved.

3.3 Project Design

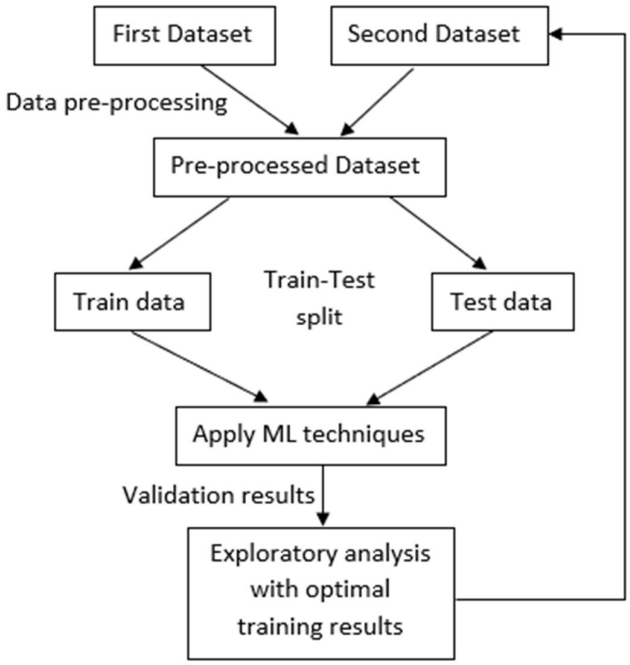

Figure 1 : Flowchart

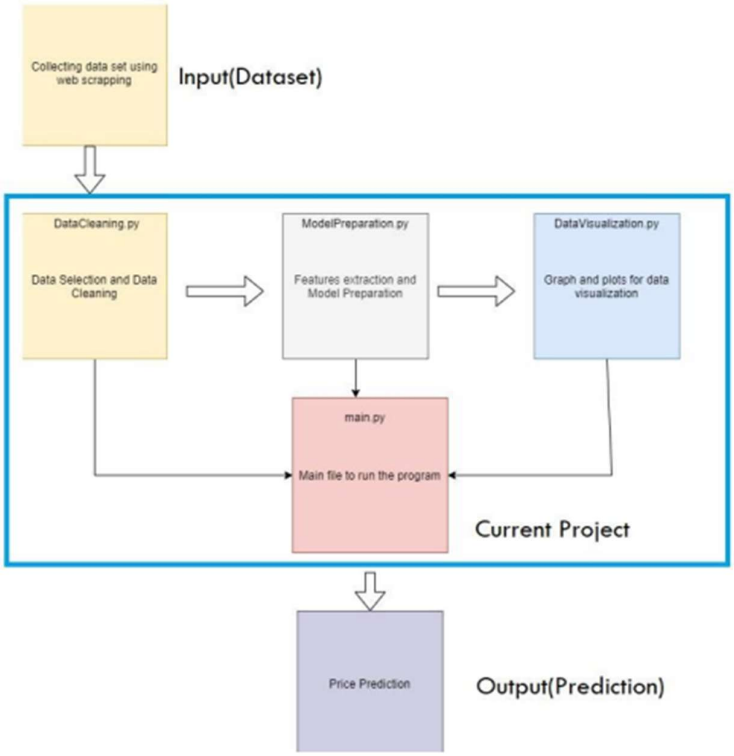

Figure 2 : Block Diagram

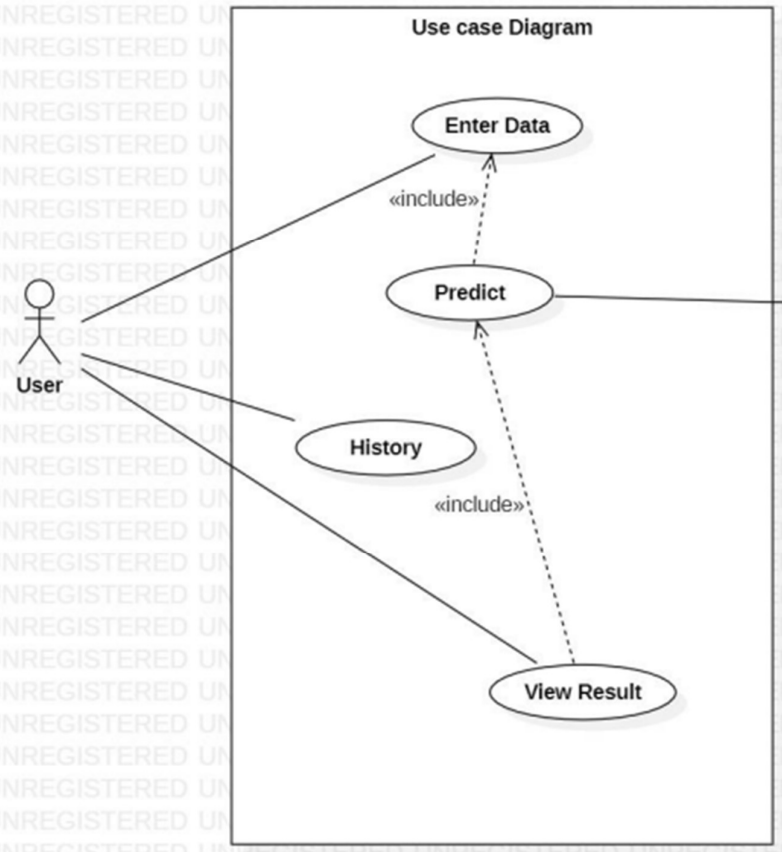

Figure 3 : Use Case Diagram

Figure 4 : DFD Level 0

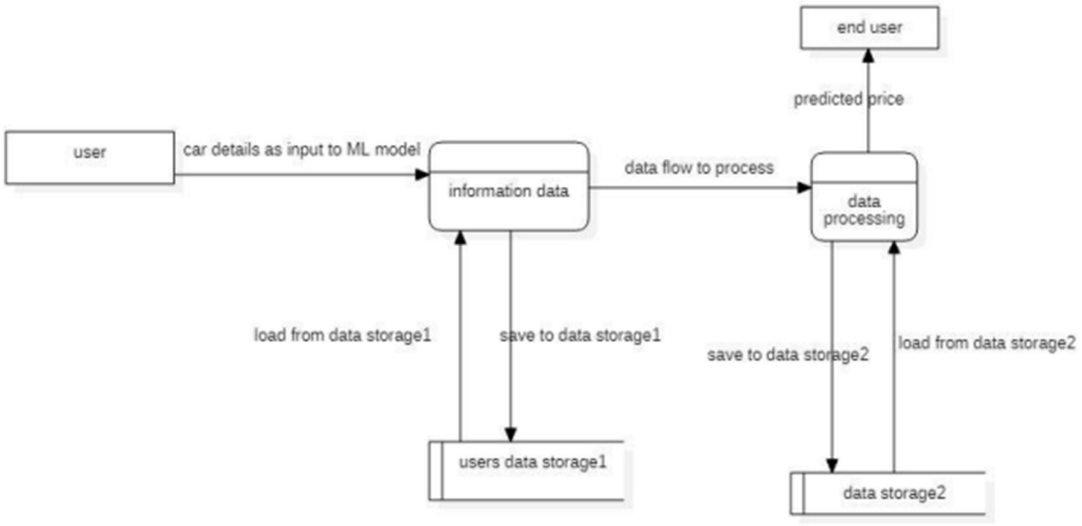

Figure 5 : DFD Level 1

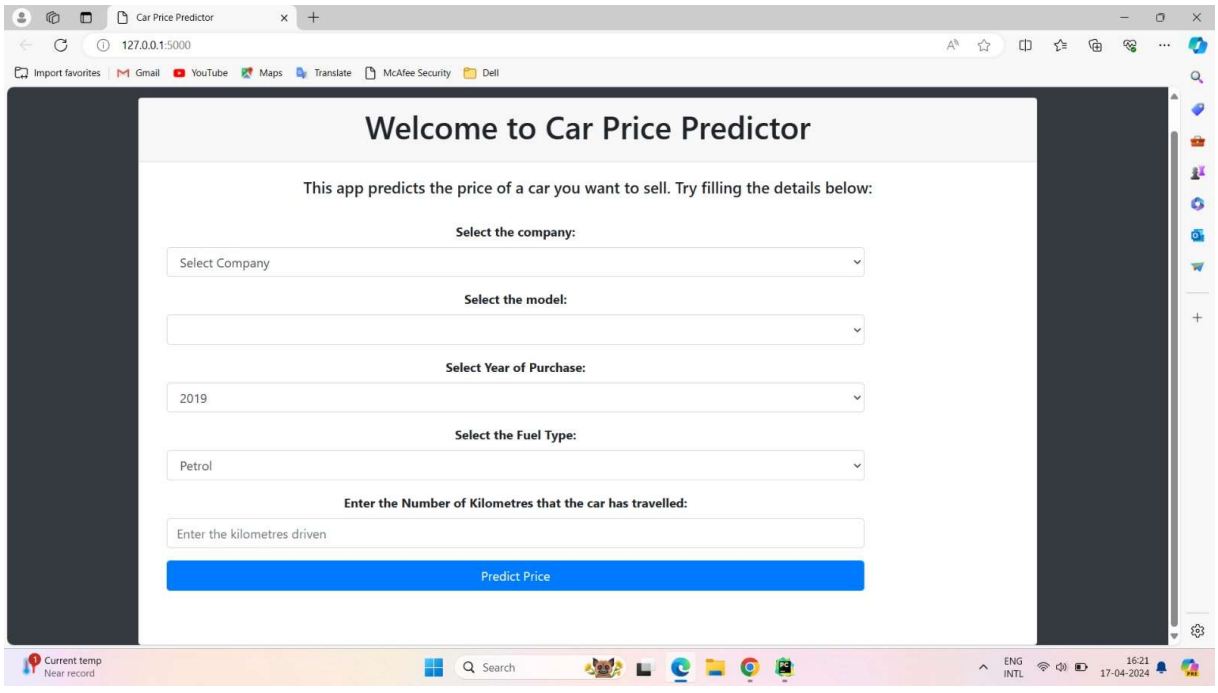

Figure 6 : Output Screenshot

Code

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import matplotlib as mpl
5 %matplotlib inline
6 mpl.style.use('ggplot')
7
```

```
1 from google.colab import files
2 uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving quikr_car.csv to quikr_car.csv
```

```
1 car=pd.read_csv('quikr_car.csv')
```

```
1 car.head()
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol

```
1 car.shape
```

```
(815, 6)
```

```
1 car.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 815 entries, 0 to 815
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        815 non-null    object
 1   company     815 non-null    object
 2   year        815 non-null    int64
 3   Price       815 non-null    int64
 4   kms_driven  815 non-null    int64
 5   fuel_type   815 non-null    object
dtypes: int64(3), object(3)
memory usage: 76.9+ KB
```

```
1 backup=car.copy()
```

```
1 car=car[car['year'].str.isnumeric()]
```

```
1 car['year']=car['year'].astype(int)
```

```
1 car=car[car['Price']!='Ask For Price']
```

```
1 car['Price']=car['Price'].str.replace(',','').astype(int)
```

```
1 car['kms_driven']=car['kms_driven'].str.split().str.get(0).str.replace(',','')
```

```
1 car=car[car['kms_driven'].str.isnumeric()]
```

```
1 car['kms_driven']=car['kms_driven'].astype(int)
```

```
1 car=car[~car['fuel_type'].isna()]
```

```
1 car.shape
2
```

```
(816, 6)
```

```
1 car['name']=car['name'].str.split().str.slice(start=0,stop=3).str.join(' ')
```

```
1 car=car.reset_index(drop=True)
```

```
1 car
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel
...	...	...	...	...	...	...
887	Ta	Tara	zest	3,10,000	NaN	NaN
888	Tata Zest XM Diesel	Tata	2018	2,60,000	27,000 kms	Diesel
889	Mahindra Quanto C8	Mahindra	2013	3,90,000	40,000 kms	Diesel
890	Honda Amaze 1.2 E i VTEC	Honda	2014	1,80,000	Petrol	NaN
891	Chevrolet Sail 1.2 LT ABS	Chevrolet	2014	1,60,000	Petrol	NaN

```
1 car.to_csv('Cleaned_Car_data.csv')
```

```
1
2 car.describe(include='all')
```

	name	company	year	Price	kms_driven	fuel_type
count	892	892	892	892	840	837
unique	525	48	61	274	258	3
top	Honda City	Maruti	2015	Ask For Price	45,000 kms	Petrol
freq	13	235	117	35	30	440

```
1 car=car[car['Price']<6000000]
```

```
1 car['company'].unique()
```

```
array(['Hyundai', 'Mahindra', 'Ford', 'Maruti', 'Skoda', 'Audi', 'Toyota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswagen',
       'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat', 'Force',
       'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'], dtype=object)
```

```
1 X=car[['name','company','year','kms_driven','fuel_type']]
2 y=car['Price']
```

```
1 X
2
```

	name	company	year	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	36000	Diesel
4	Ford Figo	Ford	2012	41000	Diesel
...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	50000	Petrol
812	Tata Indica V2	Tata	2009	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	132000	Petrol
814	Tata Zest XM	Tata	2018	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	40000	Diesel

815 rows × 5 columns

```
y.shape

```

```
(815,)
```

Applying Train Test Split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,ExtraTreesRegressor,StackingRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score,mean_absolute_error
```

Linear regression

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.5738451570697869
MAE 144851.5236418511
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=LinearRegression()
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
302
```

```
scores[np.argmax(scores)]
```

```
0.8991190499074018
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.8991190499074018
```

Ridge Regression

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr=Ridge(alpha=10)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
R2 score 0.059490142864482176
MAE 278381.9459371249
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=Ridge(alpha=10)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)

```

```
395
```

```
scores[np.argmax(scores)]
```

```
0.29224638942018066
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.6875144601893883
```

Lasso Regression

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr=Lasso(alpha=0.001)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
R2 score 0.8062313036754869
MAE 82867.1595072143
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:592: ConvergenceWarning: Objective did not converge.
  model = cd_fast.sparse_enet_coordinate_descent(
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=Lasso(alpha=0.001)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:592: ConvergenceWarning: Objec
  model = cd_fast.sparse_enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:592: ConvergenceWarning: Objec
  model = cd_fast.sparse_enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:592: ConvergenceWarning: Objec
  model = cd_fast.sparse_enet_coordinate_descent(
```

```
np.argmax(scores)
```

```
scores[np.argmax(scores)]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=Lasso(alpha=0.001)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

KNN

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr= KNeighborsRegressor(n_neighbors=3)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=KNeighborsRegressor(n_neighbors=3)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
scores[np.argmax(scores)]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=Lasso(alpha=0.001)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

Decision Tree

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr= DecisionTreeRegressor(max_depth=8)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

•••

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=DecisionTreeRegressor(max_depth=8)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)

```

```
scores[np.argmax(scores)]
```

```
0.7968601112938023
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=DecisionTreeRegressor(max_depth=8)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

SVM

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr= SVR(kernel='rbf',C=10000,epsilon=0.1)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr= SVR(kernel='rbf',C=10000,epsilon=0.1)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
484
```

```
scores[np.argmax(scores)]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=SVR(kernel='rbf',C=10000,epsilon=0.1)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

Random Forest

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr=RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)

pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr= RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```

np.argmax(scores)

```

```
scores[np.argmax(scores)]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=RandomForestRegressor(n_estimators=100,
                         random_state=3,
                         max_samples=0.5,
                         max_features=0.75,
                         max_depth=15)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.8943125673748604
```

AdaBoost

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr= AdaBoostRegressor(n_estimators=15,learning_rate=1.0)

pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
R2 score 0.22107351729171287
MAE 250660.01099106797
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr= AdaBoostRegressor(n_estimators=15,learning_rate=1.0)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
564
```

```
scores[np.argmax(scores)]
```

```
0.4956276275439062
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=AdaBoostRegressor(n_estimators=15,learning_rate=1.0)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.3682777516482736
```

Gradient Boost

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr= GradientBoostingRegressor(n_estimators=500)

pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-182a6076e53c> in <cell line: 1>()
----> 1 ohe=OneHotEncoder()
      2 ohe.fit(X[['name','company','fuel_type']])
      3 column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
      4                                      remainder='passthrough')
      5 lr= GradientBoostingRegressor(n_estimators=500)

NameError: name 'OneHotEncoder' is not defined
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr= GradientBoostingRegressor(n_estimators=500)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
507
```

```
scores[np.argmax(scores)]
```

```
0.9409603758133513
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr= GradientBoostingRegressor(n_estimators=500)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.9406240251858238
```

XgBoost

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
lr=  XGBRegressor(n_estimators=45,max_depth=5,learning_rate=0.5)

pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
R2 score 0.7102442922106702
MAE 112085.0773341641
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=XGBRegressor(n_estimators=45,max_depth=5,learning_rate=0.5)
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
507
```

```
scores[np.argmax(scores)]
```

```
0.9323895745845345
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=XGBRegressor(n_estimators=45,max_depth=5,learning_rate=0.5)
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.9323895745845345
```

Stacking

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
estimators = [
    ('rf', RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_features=0.75,max_depth=15)),
    ('gbdt',GradientBoostingRegressor(n_estimators=100,max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5))
]

lr=  StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))

```

```
R2 score 0.6886310396730786
MAE 123348.64668719996
```

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    estimators = [
    ('rf', RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_features=0.75,max_depth=15)),
    ('gbdt',GradientBoostingRegressor(n_estimators=100,max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5))
]
    lr=StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
scores[np.argmax(scores)]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
estimators = [
    ('rf', RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_features=0.75,max_depth=15)),
    ('gbdt',GradientBoostingRegressor(n_estimators=100,max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5))
]
lr=StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

Exporting the Model

```
1 import pickle
2
3 # pickle.dump(car, open('df.pkl', 'wb'))
4 pickle.dump(pipe, open('pipe.pkl', 'wb'))
5
```

```
1 car
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...

Methodology

The methodology for developing the Used Car Price Predictor project involves several key steps, including data collection, pre-processing, model development, evaluation, deployment, and ongoing maintenance. Here's a detailed breakdown of the methodology:

1. Data Collection:
- Gather comprehensive data on used cars from reliable sources, including attributes such as make, model, year, mileage, condition, region, and historical sales prices.
- Consider multiple data sources, including public datasets, APIs, web scraping from online listings, and collaboration with automotive industry partners for access to proprietary data.

2.Data Pre-processing:
- Cleanse the collected data to handle missing values, outliers, and inconsistencies using techniques such as imputation, outlier detection, and data validation.
- Perform data transformation and feature engineering to extract relevant features, encode categorical variables, and normalize numerical data for ML model compatibility.

3. Machine Learning Model Development:
- Select suitable ML algorithms for price prediction, such as decision trees, random forests, gradient boosting, or regression models, based on the nature of the data and prediction task.
- Split the pre-processed data into training and testing sets for model development and evaluation.
- Train the ML models using the training data, optimize hyperparameters through techniques like grid search or random search, and validate the models using cross-validation methods

4. Model Evaluation:
- Evaluate the trained ML models using performance metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared (R2) to assess predictive accuracy and generalization.
- Conduct sensitivity analysis and feature importance ranking to identify the most impactful variables influencing price predictions.

5. User Interface Development:
- Design and develop a user-friendly web interface or mobile application for the Used Car Price Predictor system. - Implement input forms for users to enter car details (make, model, year, mileage, condition) and receive predicted price ranges based on the trained ML models.
- Incorporate visualizations or insights to enhance user experience and decision-making.

6. Deployment:
- Deploy the trained ML models and user interface components on a suitable hosting platform or server infrastructure.
- Ensure scalability, security, and performance optimizations for handling concurrent user requests and large datasets.
- Conduct thorough testing (unit testing, integration testing, performance testing) to validate system functionality and reliability.

4. Conclusion and Future Scope

Conclusion:

The development of the Used Car Price Predictor project has been a significant endeavor aimed at addressing the pricing challenges prevalent in the used car market. Through the application of machine learning algorithms, data preprocessing techniques, and user-friendly interfaces, the project has successfully achieved its objectives of providing accurate price predictions based on vehicle specifications and market dynamics.

The system's deployment offers tangible benefits to both buyers and sellers, empowering them with data-driven insights for making informed decisions and facilitating fair transactions. The project's methodology, including data collection, preprocessing, model development, evaluation, and deployment, has laid a solid foundation for the system's functionality and reliability.

Future Scope:

While the current version of the Used Car Price Predictor system fulfils essential pricing needs, there are several avenues for future enhancement and expansion:

1. Real-time Data Integration: Incorporate real-time data streams for dynamic pricing predictions based on the latest market trends, demand fluctuations, and economic indicators.
2. Enhanced Feature Set: Include additional features such as vehicle specifications (engine size, fuel efficiency, safety ratings), sentiment analysis of customer reviews, and market sentiment analysis for more nuanced price predictions.
3. Geographical Expansion: Extend the system's coverage to encompass multiple geographical regions or global markets, adapting the models and data sources accordingly.
4. Advanced ML Techniques: Explore advanced ML techniques such as deep learning models (e.g., neural networks, recurrent neural networks) for improved predictive accuracy and pattern recognition.
5. User Experience Improvements: Continuously enhance the user interface with intuitive features, personalized recommendations, interactive visualizations, and multi-platform compatibility (web, mobile, desktop).
6. Integration with Automotive Ecosystem: Collaborate with automotive industry stakeholders, dealerships, insurance providers, and regulatory bodies to integrate additional data sources, compliance checks, and value-added services.
7. Predictive Analytics: Utilize predictive analytics for forecasting future trends, price trends, and market insights, enabling proactive decision-making and strategic planning.
8. Ethical Considerations: Address ethical considerations related to data privacy, bias mitigation, fairness in pricing predictions, and regulatory compliance, ensuring transparency and accountability in system operations.

By exploring these future scope areas, the Used Car Price Predictor system can evolve into a comprehensive and indispensable tool for stakeholders in the automotive industry, fostering transparency, efficiency, and trust in the used car market.

References

[1] https://medium.com/odscjournal/predicting-car-prices-using-machine-learning-and-data-science-52ed44abab1b

[2]https://www.divaportal.org/smash/get/diva2:1674070/FULLTEXT01.pdf

[3] https://github.com/suhasmaddali/Car-Prices-Prediction

[4] https://youtu.be/iRCaMnR_bpA?si=T0aCQnDlt37RnN25

[5] https://youtube.com/playlist?list=PLPL68eAk13fsESpD9_-2fl6zb-PFknvd8&si=hxbhh9QxJnH3ov1_