CAR PRICE PREDICTION USING MACHINE LEARNING

Submitted By:

BIDHAN CHANDRA SEN

UNIVERSITY ROLL NO: 12011222002

GUIDED BY:

Prof. Biswajit Mondal

Dr.B.C.Roy ENGINEERINGCOLLEGE

DEPARTMENT OF COMPUTERSCIENCE AND ENGINEERING

ABSTRACT

The goal of this work is to develop a model that can forecast the fair market value of used automobiles based on a number of factors, such as kms_driven, year of purchase, fuel type, present price of the car, transmission type, seller type, and Owner. In the used car industry, this model can help makers, buyers, and sellers of automobiles. When it's done, using the data that people enter, it may produce a somewhat accurate price estimate. Data science and machine learning are included in the model development process. The used dataset was taken from used vehicle listings through scraping. To attain the best accuracy, a variety of regression techniques were used in the study, including gradient boosting, random forest, support vector, XGB regression, and linear regression.

Keywords: By predicting the car price we can help the customers buy at a reasonable price.

INTRODUCTION

Because there are so many variables that influence a used car's price on the market, it can be difficult to determine whether the quoted price is reasonable. In order to make educated decisions, the goal of this project is to create machine learning models that can precisely forecast the price of a used car based on its properties. We employ and assess diverse learning techniques on a dataset comprising the selling prices of distinct brands and models. We will evaluate the effectiveness of several machine learning methods, including XGB Regressor, Random Forest Regression, Gradient Boosting Regression, and Linear Regression, and select the top performer. We will calculate the car's price based on a number of factors.
Project Objectives

In this project I have a shorted ‘prediction’ Dataset from Kaggle. We are predicting car costs, Year will be our target feature, which is a continuous attribute.

Our objective in this project is to study the given dataset of ‘Medical Insurance Cost Prediction’. We might need to pre-process the given dataset if we need to. Then, we would train 4 models viz. ‘Linear Regression classifier model’, ‘XGB Regressor model’, ‘Random Forest Regressor model’, and ‘Gradient Boosting Regression model’.

Our methodology for solving the problems in the given project is described below:

• Load the required dataset.
• Study the dataset.
• Describe the dataset.
• Visualise the dataset.
• Find out if the dataset needs to be pre-processed. It will be determined on the basis of whether the dataset has null values or outliers or any such discrepancy that might affect the output of the models to be trained.

• If the dataset is required to be pre-processed, take the necessary steps to pre-process the data.
• Find out the principal attributes for training
• Calculate the accuracy of the 4 models and find out the classification reports.
• Plot the necessary graphs.
• Use each trained model to predict the outcomes of the given test dataset.
• Choose the best model among the 4 trained models bases on the accuracy and classification reports.

Project Scope

The broad scope of ‘Car Price Prediction’ project is given below:

• The given dataset has attributes of the car.
• It is a useful project as the Classifier models can be used to quickly determine the which model is appropriate for Car Price Prediction of large datasets.
• Various banking institution can use these models and modify them according to their needs to use in their insurance. This will reduce the manual labour and time.
• Beneficiary who intend to buy car can use these trained models to check whether the beneficiary get more benefits.
• The dataset given to us is a shortened form of the original dataset from Kaggle. So, the results might have some mismatch with the real-world applications. But that can be avoided if the models are trained
Data Description

**Source of the data:** Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.

**Data Description:** The given train dataset has 301 rows and 9 columns.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Target Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car_Name</td>
<td>Car_Name</td>
<td>Non-Categorical</td>
<td>Name of the Car</td>
<td>No</td>
</tr>
<tr>
<td>Year</td>
<td>Year</td>
<td>Non-Categorical</td>
<td>Year of Purchase</td>
<td>Yes</td>
</tr>
<tr>
<td>Selling_Price</td>
<td>Selling_Price</td>
<td>Non-Categorical</td>
<td>Selling Price of the Car</td>
<td>No</td>
</tr>
<tr>
<td>Present_Price</td>
<td>Present_Price</td>
<td>Non-Categorical</td>
<td>Present Price of the Car</td>
<td>No</td>
</tr>
<tr>
<td>Kms_Driven</td>
<td>Kms_Driven</td>
<td>Non-Categorical</td>
<td>No.of kms driven by the Car</td>
<td>No</td>
</tr>
<tr>
<td>Fuel_Type</td>
<td>Fuel_Type</td>
<td>Categorical</td>
<td>Fuel_Type of the Car</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Fuel_Type</td>
<td></td>
<td>Petrol,Diesel Or CNG</td>
<td></td>
</tr>
<tr>
<td>Seller_Type</td>
<td>Seller_Type</td>
<td>Categorical</td>
<td>Seller_Type of the Car</td>
<td>Yes</td>
</tr>
<tr>
<td>Transmission</td>
<td>Transmission</td>
<td>Categorical</td>
<td>Transmission type of the Car</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmission</td>
<td></td>
<td>Manual Or Automatic</td>
<td></td>
</tr>
<tr>
<td>Owner</td>
<td>Owner</td>
<td>Categorical</td>
<td>Owner type of the Car</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Owner</td>
<td></td>
<td>First Owner ,Second Owner Or Third Owner</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows the 5 number statistics of the given dataset:

<table>
<thead>
<tr>
<th></th>
<th>Year</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>301.000000</td>
<td>301.000000</td>
<td>301.00000000</td>
<td>301.000000</td>
</tr>
<tr>
<td>mean</td>
<td>2013.627907</td>
<td>4.661296</td>
<td>7.628472</td>
<td>36947.205980</td>
</tr>
<tr>
<td>std</td>
<td>2.891554</td>
<td>5.082812</td>
<td>8.644115</td>
<td>38886.883882</td>
</tr>
<tr>
<td>min</td>
<td>2003.000000</td>
<td>0.100000</td>
<td>0.320000</td>
<td>500.000000</td>
</tr>
<tr>
<td>25%</td>
<td>2012.000000</td>
<td>0.900000</td>
<td>1.200000</td>
<td>15000.000000</td>
</tr>
<tr>
<td>50%</td>
<td>2014.000000</td>
<td>3.600000</td>
<td>6.400000</td>
<td>32000.000000</td>
</tr>
<tr>
<td>75%</td>
<td>2016.000000</td>
<td>6.000000</td>
<td>9.900000</td>
<td>48767.000000</td>
</tr>
<tr>
<td>max</td>
<td>2018.000000</td>
<td>35.000000</td>
<td>92.600000</td>
<td>500000.000000</td>
</tr>
</tbody>
</table>
Data Pre-processing

As the given dataset had Categorical and Non-categorical data mixed, we converted the categorical data into non-categorical data accordingly. We converted the binary categories into 0,1 and 2. We converted the other categorical attributes into suitable numerical values. The following table shows the conversion record:

<table>
<thead>
<tr>
<th>Non-numerical to numeric change table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
</tr>
<tr>
<td>Transmission_Type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Seller_Type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Fuel_Type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Owner_Type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

We searched for null values in our dataset and formed the following table:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Count of null value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car_Name</td>
<td>0</td>
</tr>
<tr>
<td>Year</td>
<td>0</td>
</tr>
<tr>
<td>Selling_Price</td>
<td>0</td>
</tr>
<tr>
<td>Present_Price</td>
<td>0</td>
</tr>
<tr>
<td>Kms_Driven</td>
<td>0</td>
</tr>
<tr>
<td>Fuel_Type</td>
<td>0</td>
</tr>
<tr>
<td>Seller_Type</td>
<td>0</td>
</tr>
<tr>
<td>Transmission</td>
<td>0</td>
</tr>
<tr>
<td>Owner</td>
<td>0</td>
</tr>
</tbody>
</table>
DATASET

(Methodology)

Splitting data for training and testing purpose

We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.80 which indicates we used 80% data for training purpose and 20% data for testing purpose. We will be using the same split ratio for all the models trained.

Now we will be training our required models. Our project goal requires us to train specific 4 classifier models viz.
1. Linear Regression Classifier
2. XGB Regression
3. Random Forest Classifier
4. Gradient Boosting Regression
Linear Regression Classifier

What is linear regression?
Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable’s value is called the independent variable.

# Loading the Linear Regression Model
LR = LinearRegression() [34]:
LR.fit(X_train, Y_train)

Making Prediction with Linear Regression Model:

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let’s make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

\[ y = B_0 + B_1 \times x \]

or

weight = B0 + B1 \times height

Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.
XGB Regression

XGBoost (eXtreme Gradient Boosting) is a well-known and robust machine learning algorithm often used for supervised learning tasks such as classification, regression, and ranking. It is based on gradient-boosting architecture and has gained popularity because of its high accuracy and scalability.

Random Forest Classifier

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

# Loading the Random forest Regressor Model
rf = RandomForestRegressor() rf.fit(X_train, Y_train)

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

**Gradient Boosting Regression**

Gradient boosting constructs additive regression models by sequentially fitting a simple parameterized function (base learner) to current “pseudo”-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point evaluated at the current step. It is shown that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the procedure. Specifically, at each iteration a subsample of the training data is drawn at random (without replacement) from the full training data set. This randomly selected subsample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration. This randomized approach also increases robustness against overcapacity of the base learner.

```python
# Loading the Gradient Boosting Regression Model
g = GradientBoostingRegressor()
g.fit(X_train, Y_train)
```

**How it works:**

Let’s say the output model $\hat{y}$ when fit to only 1 decision tree, is given by: $y = A_1 + (B_1 \times X) + e_1$

where, $e_1$ is the residual from this decision tree.

In gradient boosting, we fit the consecutive decision trees on the residual from the last one. So when gradient boosting is applied to this model, the consecutive decision trees will be mathematically represented as:

$e_1 = A_2 + (B_2 \times X) + e_2$

and

$e_2 = A_3 + (B_3 \times X) + e_3$

Note that here we stop at 3 decision trees, but in an actual gradient boosting model, the number of learners or decision trees is much more. Combining all three equations, the final model of the decision tree will be given...
by:

\[ y = A_1 + A_2 + A_3 + (B_1 \times x) + (B_2 \times x) + (B_3 \times x) + e_3 \]

**R Square value calculation:**

- R-squared is a statistical method that determines the goodness of fit.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a **coefficient of determination**, or **coefficient of multiple determination** for multiple regression.
- It can be calculated from the below formula:

\[
R^2 = \frac{N \sum xy - \sum x \sum y}{\sqrt{(N \sum x^2 - (\sum x)^2)(N \sum y^2 - (\sum y)^2)}}
\]

- \( R = \) The Correlation coefficient
- \( n = \) number in the given dataset
- \( x = \) first variable in the context
- \( y = \) second variable

**Mean Absolute Error or MAE**

We know that an error basically is the absolute difference between the actual or true values and the values that are predicted. Absolute difference means that if the result has a negative sign, it is ignored.

Hence, \( MAE = \) True values – Predicted values

MAE takes the **average** of this error from every sample in a dataset and gives the output.

\[ MAE = \frac{1}{N} \sum_{i=1}^{N} |y_{i,\text{pred}} - y_{i,\text{true}}| \]

(RESULT)

We trained 4 models using the 4 algorithms viz.
1. Linear Regression Classifier
2. XGB Regressor
3. Random Forest Classifier
4. Gradient Boosting Regressor

Comparison against R squared value:

Models which gives maximum R squared value is best model. Here XGB Regressor gives max value.

The 4 models had different accuracy. The comparison of the accuracies of the models are given below.

<table>
<thead>
<tr>
<th>models</th>
<th>R squared value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression Classifier</td>
<td>0.687529</td>
</tr>
<tr>
<td>XGB Regressor</td>
<td>0.888747</td>
</tr>
<tr>
<td>Random Forest Classifier</td>
<td>0.760966</td>
</tr>
<tr>
<td>Gradient Boosting Regressor</td>
<td>0.885538</td>
</tr>
</tbody>
</table>
Therefore, from the above evaluation we conclude that XGB Regression model is best for the insurance datasets.

TestDataset

We were given a test dataset for this Car Price Prediction problem. We pre-process the given test dataset in similar way we pre-processed our train dataset. The methodology followed is given below:

• Change the categorical values to numeric values using same process we used during changing values in our train dataset.
• Checking for null values.
• If null values are present, we will fill them or drop the row containing the null value based on the dataset.

GUI Interface for Joblib

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:
1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. Tk(screenName=None, baseName=None, className='Tk', useTk=1): The basic code used to create the main window of the application is: m=tkinter.Tk() where m is the name of the main window object
2. mainloop(): There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

m.mainloop()

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

Button: To add a button in your application, this widget is used. The general syntax is:
w=Button(master, option=value)
IJCRT2402362 | International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org | d85

Codes

```python
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
data = pd.read_csv('car data.csv')

1. Display Top 5 Rows Of The Dataset

```data.head()```

<table>
<thead>
<tr>
<th>Car_Name</th>
<th>Year</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
<th>Fuel_Type</th>
<th>Seller_Type</th>
<th>Transmission</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datsun</td>
<td>2011</td>
<td>2.5</td>
<td>2.75</td>
<td>10000</td>
<td>Petrol</td>
<td>Manual</td>
<td>M MT</td>
<td>First Owner</td>
</tr>
<tr>
<td>Toyota</td>
<td>2010</td>
<td>3.5</td>
<td>3.25</td>
<td>5000</td>
<td>Diesel</td>
<td>Manual</td>
<td>M MT</td>
<td>First Owner</td>
</tr>
<tr>
<td>Honda</td>
<td>2011</td>
<td>4.0</td>
<td>3.75</td>
<td>15000</td>
<td>Petrol</td>
<td>Manual</td>
<td>M MT</td>
<td>First Owner</td>
</tr>
<tr>
<td>Maruti</td>
<td>2012</td>
<td>4.5</td>
<td>4.25</td>
<td>20000</td>
<td>Diesel</td>
<td>Manual</td>
<td>M MT</td>
<td>First Owner</td>
</tr>
<tr>
<td>Tata</td>
<td>2013</td>
<td>5.0</td>
<td>4.75</td>
<td>30000</td>
<td>Petrol</td>
<td>Manual</td>
<td>M MT</td>
<td>First Owner</td>
</tr>
</tbody>
</table>

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```

```
(300, 9)
```

Number Of Rows 300
Number Of Columns 9

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()```

```
```

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

```python
#data_info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 9 columns):
# Column Non-Null Count Dtype
0 Car_Name 300 non-null object
1 Year 300 non-null object
2 Selleding_Price 300 non-null object
3 Present_Price 300 non-null object
4 Kms_Driven 300 non-null object
5 Fuel_Type 300 non-null object
6 Seller_Type 300 non-null object
7 Transmission 300 non-null object
8 Owner 300 non-null object
dtypes: float64(2), int64(2), object(5)
memory usage: 21.9 KB
```

5. Check Null Values In The Dataset

```python
#data.isnull().sum()
```


6. Get Overall Statistics About The Dataset

```python
#data.describe()
```

```plaintext
<table>
<thead>
<tr>
<th>Year</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>101.000000</td>
<td>101.000000</td>
<td>101.000000</td>
</tr>
<tr>
<td>mean</td>
<td>1953.57907</td>
<td>7082.47</td>
<td>103947.20980</td>
</tr>
<tr>
<td>std</td>
<td>3.61554</td>
<td>8644.115</td>
<td>3886.000000</td>
</tr>
<tr>
<td>min</td>
<td>2012.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>25%</td>
<td>2012.000000</td>
<td>1200.0000</td>
<td>1500.000000</td>
</tr>
</tbody>
</table>
```
6. Get Overall Statistics About The Dataset

```python
data.describe()
```

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>301.000000</td>
<td>301.000000</td>
<td>301.000000</td>
<td>301.000000</td>
</tr>
<tr>
<td>mean</td>
<td>2013.627907</td>
<td>4.661266</td>
<td>762472</td>
<td>36947.205380</td>
</tr>
<tr>
<td>std</td>
<td>2.081554</td>
<td>5.028312</td>
<td>844115</td>
<td>20386.003882</td>
</tr>
<tr>
<td>min</td>
<td>2000.000000</td>
<td>0.100000</td>
<td>0.320000</td>
<td>550.000000</td>
</tr>
<tr>
<td>25%</td>
<td>2012.000000</td>
<td>0.000000</td>
<td>1.200000</td>
<td>15000.000000</td>
</tr>
<tr>
<td>50%</td>
<td>2014.000000</td>
<td>3.000000</td>
<td>640000</td>
<td>32000.000000</td>
</tr>
<tr>
<td>75%</td>
<td>2016.000000</td>
<td>6.000000</td>
<td>900000</td>
<td>48767.000000</td>
</tr>
<tr>
<td>max</td>
<td>2018.000000</td>
<td>15.000000</td>
<td>9220000</td>
<td>500000.000000</td>
</tr>
</tbody>
</table>

7. Data Preprocessing

```python
data.head()```

<table>
<thead>
<tr>
<th>Car_Name</th>
<th>Year</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntrz</td>
<td>2014</td>
<td>3.15</td>
<td>5.59</td>
<td>27000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```python
import datetime

data['Age'] = data['Year'] - data['Year']
```

<table>
<thead>
<tr>
<th>Car_Name</th>
<th>Year</th>
<th>Selling_Price</th>
<th>Present_Price</th>
<th>Kms_Driven</th>
<th>Fuel_Type</th>
<th>Seller_Type</th>
<th>Transmission</th>
<th>Owner</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntrz</td>
<td>2014</td>
<td>3.15</td>
<td>5.59</td>
<td>27000</td>
<td>Petrol</td>
<td>Dealer</td>
<td>Manual</td>
<td>First Owner</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 8. Store Feature Matrix in X and Response(Target) In Vector y

```python
X = data.drop(['car_name', 'selling_price'], axis=1)
y = data['selling_price']
```

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

```python
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, random_state=42)
```

### 10. Import The models
10. Import The models

```python
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

11. Model Training

```python
# lr = LinearRegression()
# lr.fit(X_train,y_train)
# rf = RandomForestRegressor()
# rf.fit(X_train,y_train)
# gbr = GradientBoostingRegressor()
# gbr.fit(X_train,y_train)
# xgb = XGBRegressor()
# xgb.fit(X_train,y_train)
```

12. Prediction on Test Data

```python
# y_pred1 = lr.predict(X_test)
# y_pred2 = rf.predict(X_test)
# y_pred3 = gbr.predict(X_test)
# y_pred4 = xgb.predict(X_test)
```

13. Evaluating the Algorithm

```python
from sklearn import metrics
```
13. Evaluating the Algorithm

```python
from sklearn import metrics

corel1 = metrics.r2_score(y_test,y_pred)
corel2 = metrics.r2_score(y_test,y_pred)
corel3 = metrics.r2_score(y_test,y_pred)
corel4 = metrics.r2_score(y_test,y_pred)

print('corel1,corel2,corel3,corel4')
```

```python
final_data = pd.DataFrame({'Models':['LR','RF','GBR','XG'],'R2_SCORE':corel1,corel2,corel3,corel4})

# final_data[final_data['R2_SCORE'].max()]
```

14. Save Model using Joblib

```python
xg = XGBRegressor()
xg_final = xg.fit(X,y)

import joblib

joblib.dump(xg_final, 'car_price_predictor')
```

```python
['car_price_predictor]
```

15. Prediction on New Data

```python
import pandas as pd
data_new = pd.DataFrame({'
'Pricen_Present':1.50,
'Fuel_Type':1.08,
'Owner_Type':0,
'Transmission':0,
'Kms_Driven':100000,
'Age':0,

','_index':4')

model.predict(data_new)
```
GUI Creation of the Project

```python
from tkinter import *

def show_entry_fields():
    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(
        {'Present_Price': [p1],
         'Km_Driven': [p2],
         'Fuel_Type': [p3],
         'Seller_Type': [p4],
         'Transmission': [p5],
         'Owner': [p6],
         '"age": [p7],
    }

    result=model.predict(data_new)

    resultfloat=result[0]

    resultint=int(resultfloat)

    Label(root, text="Car Purchase amount\:").grid(row=6)
    Label(root, text=resultint).grid(row=7)
```

```python
def main():
    root = Tk()
    root.title("GUI for Car Price Prediction")
    root.geometry("600x250")

    Label(root, text="Present_Price: ").grid(row=0)
    Label(root, text=car_price).
    e1 = Entry(root)
    e1.grid(row=0, column=1)

    Label(root, text="Km_Driven: ").grid(row=1)
    Label(root, text=km_driven).
    e2 = Entry(root)
    e2.grid(row=1, column=1)

    Label(root, text="Fuel_Type: ").grid(row=2)
    Label(root, text=fuel_type).
    e3 = Entry(root)
    e3.grid(row=2, column=1)

    Label(root, text="Seller_Type: ").grid(row=3)
    Label(root, text=seller_type).
    e4 = Entry(root)
    e4.grid(row=3, column=1)

    Label(root, text="Transmission: ").grid(row=4)
    Label(root, text=transmission).
    e5 = Entry(root)
    e5.grid(row=4, column=1)

    Label(root, text="Owner: ").grid(row=5)
    Label(root, text=owner).
    e6 = Entry(root)
    e6.grid(row=5, column=1)

    Label(root, text="Age: ").grid(row=6)
    Label(root, text=age).
    e7 = Entry(root)
    e7.grid(row=6, column=1)

    button = Button(root, text="Predict", command=show_entry_fields)
    button.grid(row=7)

    root.mainloop() 
```

```python
if __name__ == "__main__":
    main()
```
CONCLUSION
Sales of used automobiles are rising globally due to rising new car prices and consumers' inability to afford them. Consequently, there is a pressing need for a Used Car Price Prediction system that uses a range of features to efficiently assess the car's worthiness. The suggested method will assist in estimating used automobile prices with greater accuracy. In this work, four distinct machine learning algorithms—linear regression, random forests, Gradient Boosting and XGB Regression—are compared.

Acknowledgement
I take this opportunity to express my profound gratitude and deep regards to my faculty, supervisor Prof. Biswajit Mondal for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.
I am obliged to my project mentor for the valuable information provided by him in his respective fields. I am grateful for his cooperation during the period of my assignment.

Bidhan Chandra Sen

Future Scope For Improvement

● Various car company can use these models and modify them according to their need.
● Beneficiary who intend to buy a car can use these trained models to check whether they get best benefit. The trained models would be required to be implemented in a platform or interface easily accessible as well as with an easy GUI.

BIBLIOGRAPHY

https://ieeexplore.ieee.org/document/9696839