

CAR RESALE VALUE PREDICTION

Project report

Submitted by

ABISHEAK HEMARENGAN P

DHAKSHINESH E B

ABISHEK R

GOWRI KISHORE

From

BANANRI AMMAN INSTITUTE OF TECHNOLOGY



November 2022

ABSTRACT

To be able to predict used cars market value can help both buyers and sellers. There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it's a big corner to pay too much or sell less than its market value. In this Project, we are going to predict the Price of Used Cars using various features like year, model type, brand, fuel type, kilo-meter. Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and its 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. To overcome this problem we have developed a model which will be highly effective. Gradient boosting Regressor is used because calculates the difference between the current prediction and the known correct target 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.

Keyword: Gradient boosting regressor, Machine Learning, used car value predication

TABLE OF CONTENTS

| Chapter | Content | Page No |
|----------------|---------------------------------------------------------------|----------------|
| 1. | Introduction | 4 |
| | 1.1 Problem statement | 4 |
| | 1.2 Project goal | 4 |
| | 1.3 Machine Learning | 5 |
| | 1.4 Gradient boosting regressor | 5 |
| 2. | Literature survey | 6 |
| | 2.1 Car resale prediction system | 6 |
| | 2.2 Vehicle resale price prediction using machine learning | 6 |
| 3. | Module description | 7 |
| | 3.1 Collect dataset | 7 |
| | 3.1 Training and testing phase | 12 |
| 4. | System specification | 19 |
| | 4.1 Hardware specification | 19 |
| | 4.2 software specification | 19 |
| | 4.3 software description | 19 |
| 5. | Output | 21 |
| 6. | Conclusion | 23 |

CHAPTER 1

1. INTRODUCTION:

Almost everyone wants their own car these days, but because of factors like affordability or economic conditions, many prefer to opt for pre-owned cars. Accurately predicting used car prices requires expert knowledge due to the nature of their dependence on a variety of factors and features. Used car prices are not constant in the market, both buyers and sellers need an intelligent system that will allow them to predict the correct price efficiently. In this intelligent system, the most difficult problem is the collection of the dataset which contains all important elements like the manufacturing year of the car, its gas type, its condition, miles driven, horsepower, doors, number of times a car has been painted, customer reviews, the weight of the car, etc. It is necessary to pre-process and transform collected data in the proper format prior to feeding it directly to the data mining model. As a first step, the dataset was statistically analysed and plotted. Missing, duplicated, and null values were identified and dealt with. Features were chosen and extracted using correlation matrices. To build an efficient model, the most correlated features were retained, and others were discarded. This prediction problem can be considered a regression problem since it belongs to the supervised learning domain.

1.1 PROBLEM STATEMENT:

It is easy for any company to price their new cars based on the manufacturing and marketing cost it involves. But when it comes to a used car it is quite difficult to define a price because it involves it is influenced by various parameters like car brand, manufactured year and etc. The goal of our project is to predict the best price for a pre-owned car in the Indian market based on the previous data related to sold cars using machine learning.

1.2 PROJECT GOAL:

Cars are more than just a utility for many. We all have different tastes when it comes to owning a car or at least when thinking of owning one. Some fit in our budget and some luxury brands are heavy on our pockets. But that should not stop us from owning it, at least used ones. The goal of this project to predict the costs of used cars to enable the buyers to make informed purchase using the data collected from various sources and distributed across various locations in India.

1.3 MACHINE LEARNING:

The goal of machine learning (ML) is to help a computer learn without being explicitly instructed to do so by means of mathematical models of data. Artificial intelligence (AI) is a subset of machine learning. Data is analysed using algorithms to identify patterns, which are then used to create predictive models. Like humans, machine learning becomes more accurate with more data and experience. With machine learning, you can adapt to situations where data is constantly changing, the nature of the request or task is shifting, or coding a solution isn't feasible.

1.4 GRADIENT BOOSTING REGRESSOR:

Gradient boosting is one of the most popular machine learning algorithms for tabular datasets. It is powerful enough to find any nonlinear relationship between your model target and features and has great usability that can deal with missing values, outliers, and high cardinality categorical values on your features without any special treatment.

CHAPTER 2

2. LITERATURE SURVEY:

2.1 CAR RESALE PREDICTION SYSTEM

Author: Dhvani Nimbark, Akshat Patel, Sejal Thakkar - 2021

Used car resale market in India was marked at 24.2 billion US dollars in 2019. Due to the huge requirement of used cars and lack of experts who can determine the correct valuation, there is an utmost need of bridging this gap between sellers and buyers. This project focuses on building a system that can accurately predict a resale value of the car based on minimal features like kms driven, year of purchase etc. without manual or human interference and hence it remains unbiased.

2.2 VEHICLE RESALE PRICE PREDICTION USING MACHINE LEARNING

Author: B.Lavanya, Sk.Reshma, N.Nikitha, M.Namitha

The production of vehicles has been consistently expanding in the previous decade, with more than 70 million traveler's vehicles being delivered in the year 2016. This has brought about the trade-in vehicle market, which all alone has become a roaring industry. The new approach of online gateways has worked with the requirement for both the client and the merchant to be better educated about the patterns and examples that decide the worth of a pre-owned vehicle on the lookout. Utilizing Machine Learning Algorithms like Linear Regression, Multiple Regression. we will attempt to foster a factual model which will actually want to anticipate the cost of a pre-owned vehicle, in light of past shopper information and a given arrangement of highlights. We will likewise be contrasting the forecast precision of these models to decide the ideal one.

CHAPTER 3

3. MODULE DESCRIPTION:

3.1 COLLECT DATASET:

Machine Learning has become a tool used in almost every task that requires estimation. So we need to build a model to estimate the price of used cars. The model should take car-related parameters and output a selling price. On sprint-1 the selling price of a used car depends on certain features datasets are collected from different open sources like kaggle.com, data.gov, UCI machine learning repository, the dataset which contains a set of features through which the resale price of the car can be identified is to be collected as

- price
- vehicle Type
- year Of Registration
- gearbox
- model
- kilo meter
- month Of Registration
- fuel Type
- brand
- not Repaired Damage

ML is a data hunger technology, it depends heavily on data, without data, it is impossible. It is the most crucial aspect that makes algorithm training possible. Collects Data, Import necessary packages, Pre-process images, and passes on to Network Model and Saves Model Weights. The libraries can be imported,

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import r2_score
```



```
1 df = pd.read_csv('data/cars.csv')
2 df.head()
```

| id | name | year | month | price | km | brand | gearbox | fuel | engine | transmission | color | status | location | dealer | price | price | price | price | price |
|----|------------|------|-------|---------|--------|-------|---------|--------|--------|--------------|-------|--------|----------|--------|---------|---------|---------|---------|---------|
| 1 | 2010-01-01 | 2010 | 1 | 1000000 | 100000 | BMW | Manual | Petrol | 2000 | Automatic | Black | Good | Delhi | BMW | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 |
| 2 | 2010-01-01 | 2010 | 1 | 1000000 | 100000 | BMW | Manual | Petrol | 2000 | Automatic | Black | Good | Delhi | BMW | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 |
| 3 | 2010-01-01 | 2010 | 1 | 1000000 | 100000 | BMW | Manual | Petrol | 2000 | Automatic | Black | Good | Delhi | BMW | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 |
| 4 | 2010-01-01 | 2010 | 1 | 1000000 | 100000 | BMW | Manual | Petrol | 2000 | Automatic | Black | Good | Delhi | BMW | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 |
| 5 | 2010-01-01 | 2010 | 1 | 1000000 | 100000 | BMW | Manual | Petrol | 2000 | Automatic | Black | Good | Delhi | BMW | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 |

Pre-Process The Data:

Pre-processing the dataset that includes:

- Handling the null values.
- Handling the categorical values if any.

- Normalize the data if required.
- Identify the dependent and independent variables.

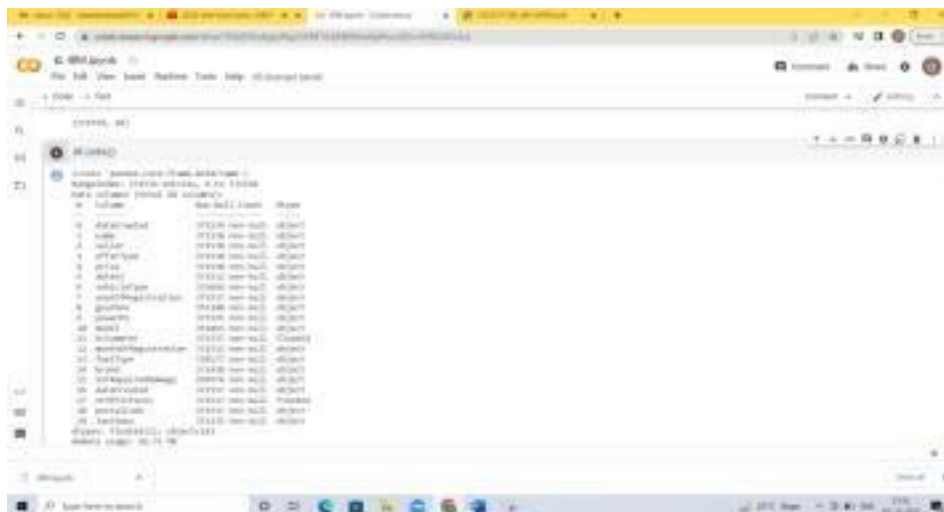
Data cleaning and wrangling methods are applied on the *used cars* data file. Before making data cleaning, some explorations and data visualizations were applied on data set. This gave some idea and guide about how to deal with missing values and extreme values. After data cleaning, data exploration was applied again in order to understand cleaned version of the data.

```
df = pd.read_csv("/content/drive/MyDrive/Colab  
Notebooks/autos.csv") df.head()
```

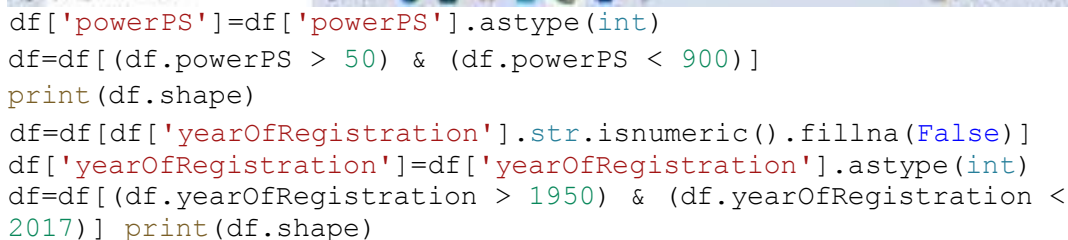
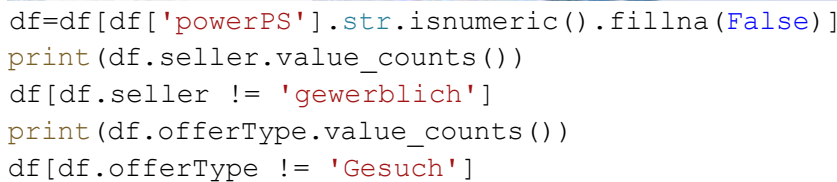
[illegible]

```
print(df.shape)
(371539, 20)
```

```
df.info()
```



```
df['powerPS'].unique()
```

```
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen',
'postalCode', 'dateCreated'], axis='columns', inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 308923 entries, 1 to 371538
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   seller                                308923 non-null object
1   offerType                             308923 non-null object
2   price                                 308923 non-null object
3   vehicleType                           297510 non-null object
4   yearOfRegistration                    308923 non-null int64
5   gearbox                               303629 non-null object
6   powerPS                               308923 non-null int64
7   model                                 297134 non-null object
8   kilometer                             308923 non-null float64
9   monthOfRegistration                   308923 non-null object
10  fuelType                              293046 non-null object
11  brand                                 308923 non-null object
12  notRepairedDamage                     265507 non-null object
dtypes: float64(1), int64(2), object(10)
memory usage: 33.0+ MB
```

```
new_df=df.copy()
new_df = new_df.drop_duplicates(['price', 'vehicleType',
'yearOfRegistration',
'gearbox', 'powerPS', 'model', 'kilometer', 'monthOfRegistration',
'fuelType',
'notRepairedDamage'])
new_df.gearbox.replace(('manuell', 'automatik'), ('manual',
'automatic' ), inplace=True)
new_df.fuelType.replace(('benzin', 'andere', 'elektro'), ('petrol',
'others', 'electric'), inplace=True)
new_df.notRepairedDamage.replace(('ja', 'nein'), ('Yes', 'No'),
inplace= True)
new_df.vehicleType.replace(('kleinwagen', 'cabrio', 'kombi',
'andere'), ('small car', 'convertible', 'combination', 'others'),
inplace=True)
new_df['price'].unique()
```

```
new_df['price'].unique()
```

```
array(['18300', '9800', '1500', ..., '18429', '24895', '10985'],
      dtype=object)
```

```
new_df['price']=new_df['price'].astype(int)
```

```

new_df = new_df[(new_df.price >= 100) & (new_df.price <=
150000)] new_df['fuelType'].fillna (value='not-declared',
inplace=True) new_df['gearbox'].fillna (value='not-declared',
inplace=True)

new_df['notRepairedDamage'].fillna (value='not-declared',
inplace=True)

new_df['vehicleType'].fillna (value='not-declared',
inplace=True) new_df['model'].fillna (value='not-declared',
inplace=True)
new_df['kilometer']=new_df['kilometer'].astype(int)
new_df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 278363 entries, 1 to 371538
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   seller                                278363 non-null  object
1   offerType                            278363 non-null  object
2   price                                278363 non-null  int64
3   vehicleType                          278363 non-null  object
4   yearOfRegistration                   278363 non-null  int64
5   gearbox                              278363 non-null  object
6   powerPS                              278363 non-null  int64
7   model                                278363 non-null  object
8   kilometer                            278363 non-null  int64
9   monthOfRegistration                  278363 non-null  object
10  fuelType                             278363 non-null  object
11  brand                                 278363 non-null  object
12  notRepairedDamage                   278363 non-null  object
dtypes: int64(4), object(9)
memory usage: 29.7+ MB

```

```
new_df.head()
```

```

new_df.head()

```

| | seller | offerType | price | vehicleType | yearOfRegistration | gearbox | powerPS | model | kilometer | monthOfRegistration | fuelType | brand | notRepairedDamage |
|---|--------|-----------|-------|-------------|--------------------|-----------|---------|--------------|-----------|---------------------|----------|------------|-------------------|
| 1 | privat | Angelot | 18800 | coupe | 2011 | manual | 180 | not-declared | 120000 | 8 | diesel | audi | Yes |
| 2 | privat | Angelot | 8800 | suv | 2004 | automatic | 103 | grand | 120000 | 8 | diesel | jeep | not-declared |
| 3 | privat | Angelot | 1500 | small car | 2001 | manual | 75 | golf | 150000 | 8 | petrol | volkswagen | No |
| 4 | privat | Angelot | 3000 | small car | 2008 | manual | 60 | fiata | 90000 | 7 | diesel | skoda | No |
| 5 | privat | Angelot | 650 | broadside | 1906 | manual | 102 | 3er | 150000 | 10 | petrol | bmw | Yes |

3.2 TRAINING AND TESTING PHASE:

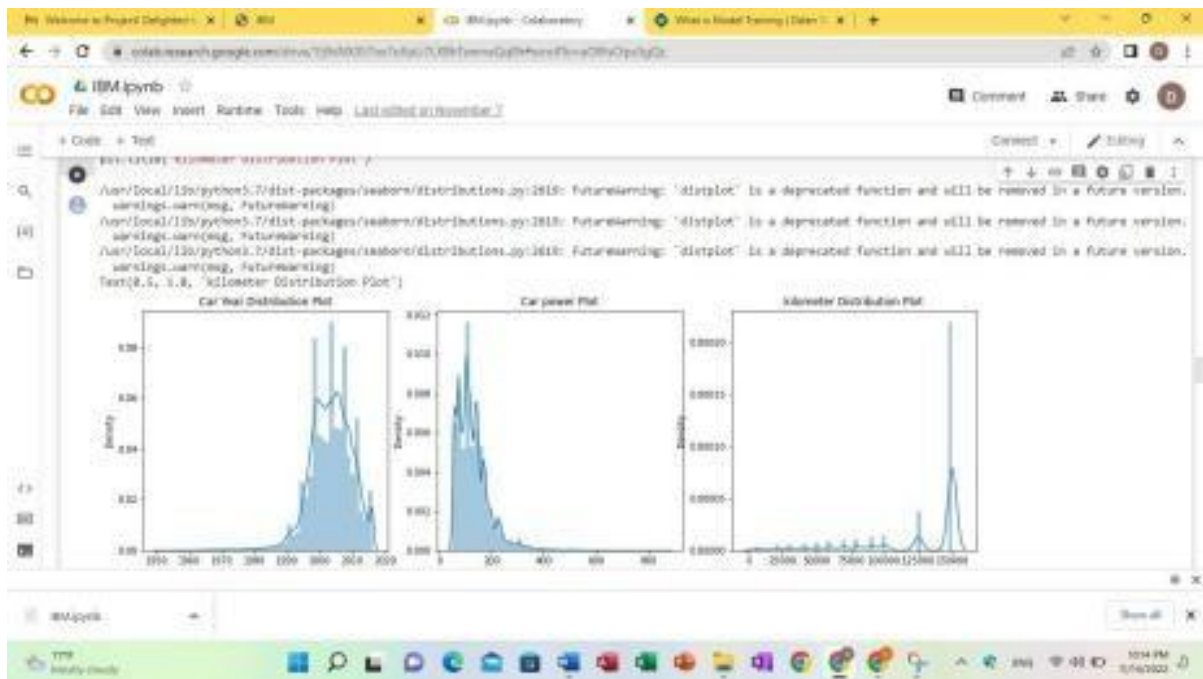
A training model is a dataset that is used to train an algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training is the process of feeding an algorithm with data to help identify and learn good values for all attributes involved.

```
import seaborn as sns
from matplotlib import *
import sys
from pylab import *
plt.figure(figsize=[11,5])
sns.distplot(new_df['price'])
```

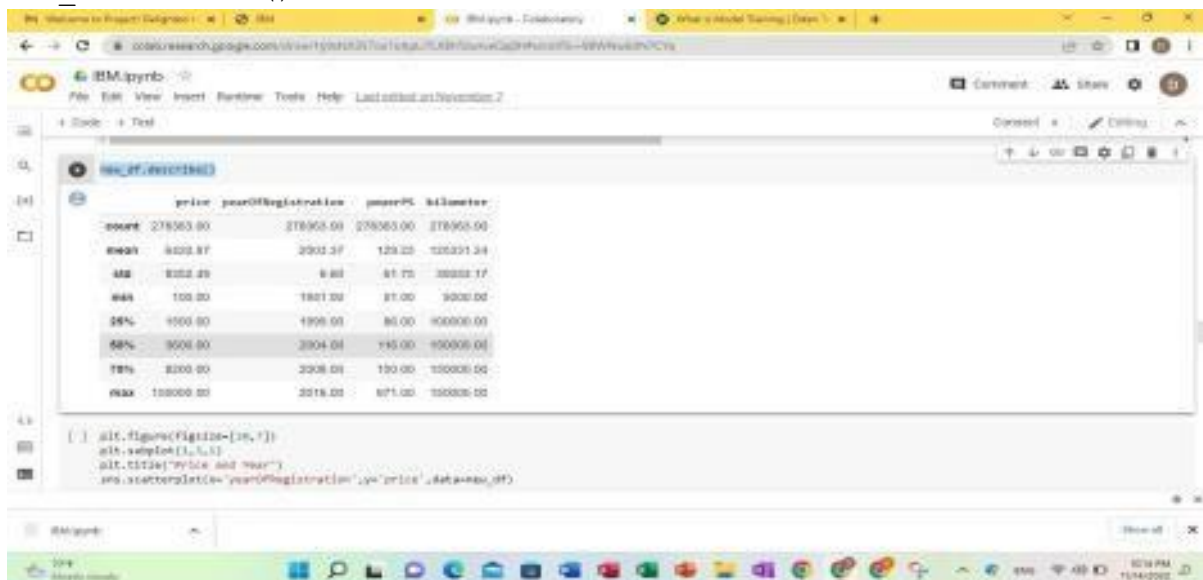


```
plt.figure(figsize=[17,5])
plt.subplot(1,3,1)
sns.distplot(new_df['yearOfRegistration'])
plt.title('Car Year Distribution Plot')

plt.subplot(1,3,2)
sns.distplot(new_df['powerPS'])
plt.title('Car power Plot')
```



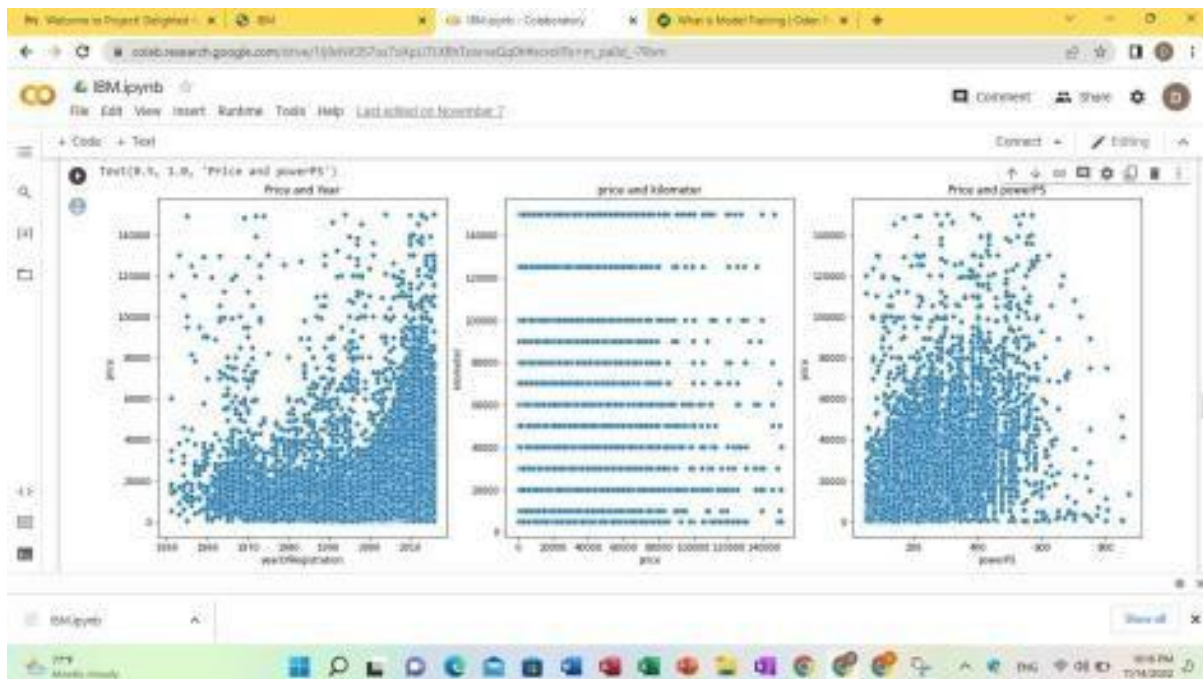
```
new_df.describe()
```



```
plt.figure(figsize=[20,7])
plt.subplot(1,3,1)
plt.title("Price and Year")
sns.scatterplot(x='yearOfRegistration',y='price',data=new_df)
```

```
plt.subplot(1,3,2)
plt.title("price and kilometer")
sns.scatterplot(x='price',y='kilometer',data=new_df)
```

```
plt.subplot(1,3,3)
sns.scatterplot(y='price',x='powerPS',data=new_df)
plt.title("Price and powerPS")
```

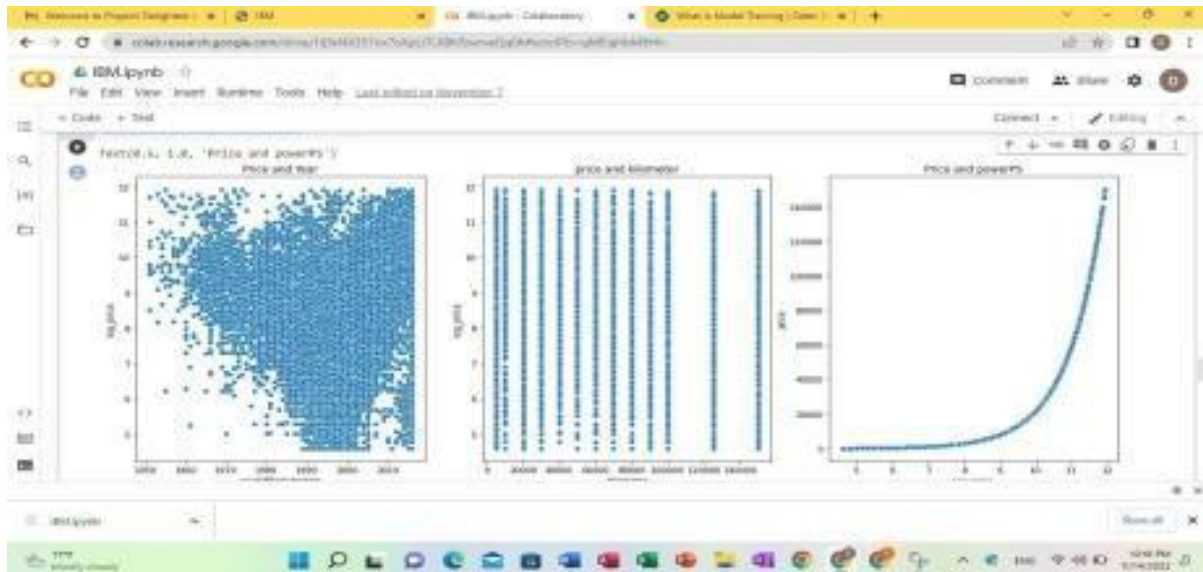
```
log_price = np.log(new_df['price'])
new_df['log_price'] = log_price
new_df.head()
```

| | seller | offerType | price | vehicleType | yearOfRegistration | gearbox | powerPS | model | kilometer | monthOfRegistration | fuelType | brand | notRepairedDamage |
|---|--------|-----------|-------|-------------|--------------------|-----------|---------|--------------|-----------|---------------------|----------|------------|-------------------|
| 1 | privat | Angebot | 18300 | coupe | 2011 | manual | 190 | not-declared | 125000 | 5 | diesel | audi | yes |
| 2 | privat | Angebot | 9900 | suv | 2004 | automatic | 163 | grand | 125000 | 8 | diesel | jeep | not-declared |
| 3 | privat | Angebot | 1500 | small car | 2001 | manual | 75 | golf | 150000 | 6 | petrol | volkswagen | No |
| 4 | privat | Angebot | 3900 | small car | 2008 | manual | 69 | tabia | 90000 | 7 | diesel | skoda | No |
| 5 | privat | Angebot | 650 | limousine | 1995 | manual | 102 | 3er | 150000 | 10 | petrol | bmw | yes |

```
plt.figure(figsize=[20,7])
plt.subplot(1,3,1)
plt.title("Price and Year")
sns.scatterplot(x='yearOfRegistration',y='log_price',data=new_df)

plt.subplot(1,3,2)
plt.title("price and kilometer")
sns.scatterplot(x='kilometer',y='log_price',data=new_df)

plt.subplot(1,3,3)
sns.scatterplot(y='price',x='log_price',data=new_df)
plt.title("Price and powerPS")
```



```
new_df= new_df.drop(['price'],axis=1)
new_df['monthOfRegistration']=new_df['monthOfRegistration'].ast
ype(int)labels= ['gearbox', 'notRepairedDamage', 'model',
'brand', 'fuelType','vehicleType']
mapper={}
for i in labels:
    mapper[i] =LabelEncoder()
    mapper[i].fit(new_df[i])
    tr=mapper[i].transform(new_df[i])
    np.save(str('classes'+i+'.numpy'), mapper[i].classes_)
    print(i, ":",mapper[i])
    new_df.loc[:, i+'_labels'] = pd.Series (tr, index=new_df.index)
```

```
labeled
=new_df[ ['log_price','yearOfRegistration','powerPS','kilom
eter','monthOfRegistration']
+ [x+"_labels" for x in labels]]
print(labeled.columns)
```

```
gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
Index(['log_price', 'yearOfRegistration', 'powerPS', 'kilometer',
'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
'model_labels', 'brand_labels', 'fuelType_labels',
'vehicleType_labels'],
dtype='object')
```

```
plt.figure(figsize=[15,7])
sns.heatmap(new_df.corr(), annot=True)
```



```
Y = labeled.iloc[:,0].values
X = labeled.iloc[:,1:].values
Y = Y.reshape(-1,1)
from sklearn.model_selection import
train_test_split,cross_val_scoreX_train, X_test, Y_train,
Y_test = train_test_split(X,Y,test_size=0.3,random_state=3)

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2_score
regressor= RandomForestRegressor (n_estimators=1000, max_depth=10,
random_state=34)

regressor.fit(X_train, np.ravel (Y_train, order='C'))
y_pred=regressor.predict(X_test)
print(r2_score (Y_test,y_pred))
y_pred=regressor.predict(X_test)
print(r2_score (Y_test,y_pred))
df_ev = pd.DataFrame(np.exp(y_pred), columns=['Predicted Price'])

# We can also include the Actual price column in that data frame
(sowecan manually compare them)
#Y_test=Y_test.reset_index(drop=True)
df_ev['Actual Price'] = np.exp(Y_test)

# we can calculate the difference between the targets and the
predictions
df_ev['Residual'] = df_ev['Actual Price'] - df_ev['Predicted
Price']df_ev['Difference%'] =
np.absolute(df_ev['Residual']/df_ev['Actual Price']*100)

pd.set_option('display.float_format', lambda x: '%.2f' % x)
df_ev.sort_values(by=['Difference%'])
```



```
df_ev.tail(5)
```

| | Predicted Price | Actual Price | Residual | Difference% |
|-------|-----------------|--------------|----------|-------------|
| 83504 | 4946.32 | 5790.00 | 843.68 | 14.57 |
| 83505 | 4177.92 | 5200.00 | 1022.08 | 19.66 |
| 83506 | 11025.04 | 12499.00 | 1473.96 | 11.79 |
| 83507 | 7967.92 | 9800.00 | 1832.08 | 18.69 |
| 83508 | 564.48 | 400.00 | -164.48 | 41.12 |

```
from sklearn.linear_model import LinearRegression lr =  
LinearRegression()  
lr.fit(X_train,Y_train)  
y_pred_lr = lr.predict(X_test)  
r_squared = r2_score(Y_test,y_pred_lr)  
print("R_squared :",r_squared)
```

```
from sklearn.ensemble import GradientBoostingRegressor gbt  
= GradientBoostingRegressor()  
gbt.fit(X_train,Y_train)  
y_pred_gbt = gbt.predict(X_test)  
r_squared = r2_score(Y_test,y_pred_gbt)  
print("R_squared :",r_squared)
```

```
df_ev = pd.DataFrame(np.exp(y_pred_gbt), columns=['Predicted  
Price'])df_ev['Actual Price'] = np.exp(Y_test)  
df_ev['Residual'] = df_ev['Actual Price'] - df_ev['Predicted  
Price']df_ev['Difference%'] =  
np.absolute(df_ev['Residual']/df_ev['Actual Price']*100)  
pd.set_option('display.float_format', lambda x: '%.2f' % x)  
df_ev.sort_values(by=['Difference%'])
```

```
df_ev.tail(5)
```

The screenshot shows a Jupyter Notebook interface with a table of model results and a code cell. The table has five columns: Predicted Price, Actual Price, Residual, and Difference. The code cell contains the following Python code:

```
if ev.tail(5):  
  
    Predicted Price Actual Price Residual Difference  
    83564 553.487 5780.00 225.05 4.06  
    83565 9854.99 5280.00 1580.81 25.79  
    83566 9025.38 12490.00 2669.62 29.36  
    83567 5083.75 3880.00 1245.27 12.72  
    83568 543.32 480.00 -240.32 83.08  
  
[ ] filename = 'resale_model.sav'  
    pickle.dump(gbt, open(filename, 'wb'))
```

The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 10:27 PM on 11/16/2022.

```
filename = 'resale_model.sav'  
pickle.dump(gbt, open(filename, 'wb'))
```

CHAPTER 4

4. SYSTEM SPECIFICATION

4.1 HARDWARE SPECIFICATION

- Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2cores, 2 threads per core), 8 GB of RAM
- Disk space: 320 GB
- Operating systems: Windows® 10, macOS*, and Linux*

4.2 SOFTWARE SPECIFICATION

- Python 3.7.4(64-bit) or (32-bit)
- HTML, CSS, javascript
- Flask 1.1.1
- Jupyter Notebook
- Windows 10 64 –bit

4.3 SOFTWARE DESCRIPTION:

FLASK:

Flask is an open source web framework which offers us with the tools, library resources needed to create a web application. Flask is a microweb framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

PANDAS:

Data analysis and related manipulation of tabular data in Data frames are the major uses of Pandas. Data can be imported into Pandas from a variety of filetypes, including Microsoft Excel, JSON, Parquet, SQL database tables, and comma-separated values. Data wrangling, data cleaning, and other operations like merging, restructuring, and choosing are all possible with Pandas. Many of the R programming language's established functionality for working with data frames were brought into Python with the introduction of pandas. The NumPy library, which is focused on effectively working with arrays rather than the characteristics of working with Data frames, is the foundation upon which the Panda library is constructed.

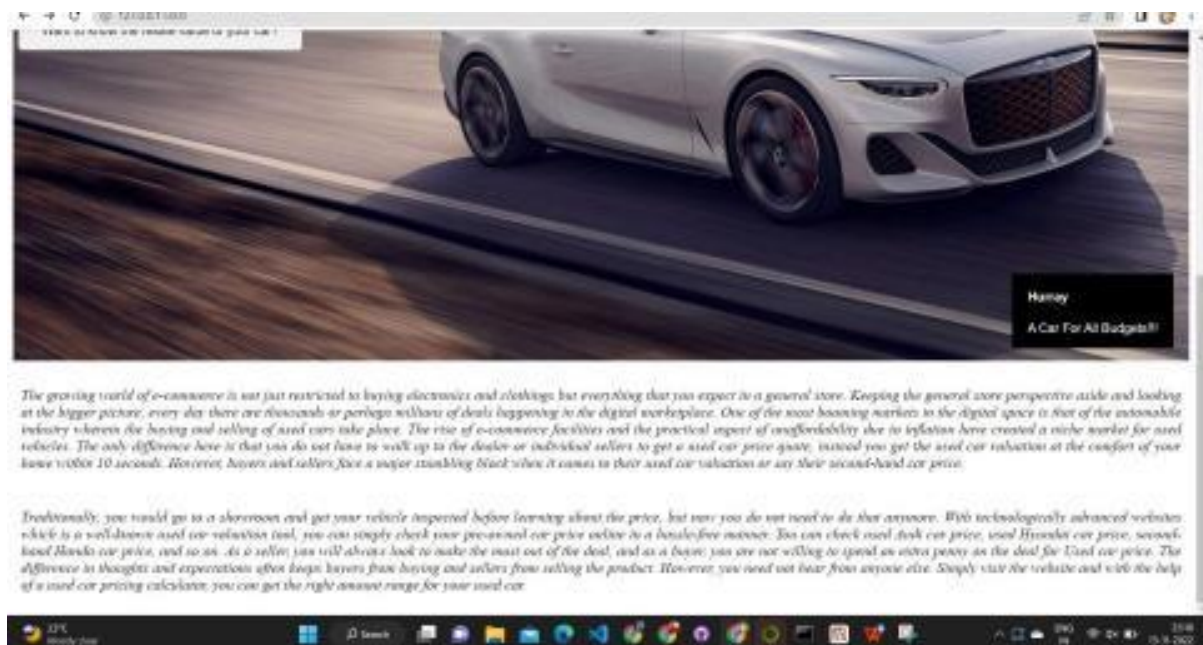
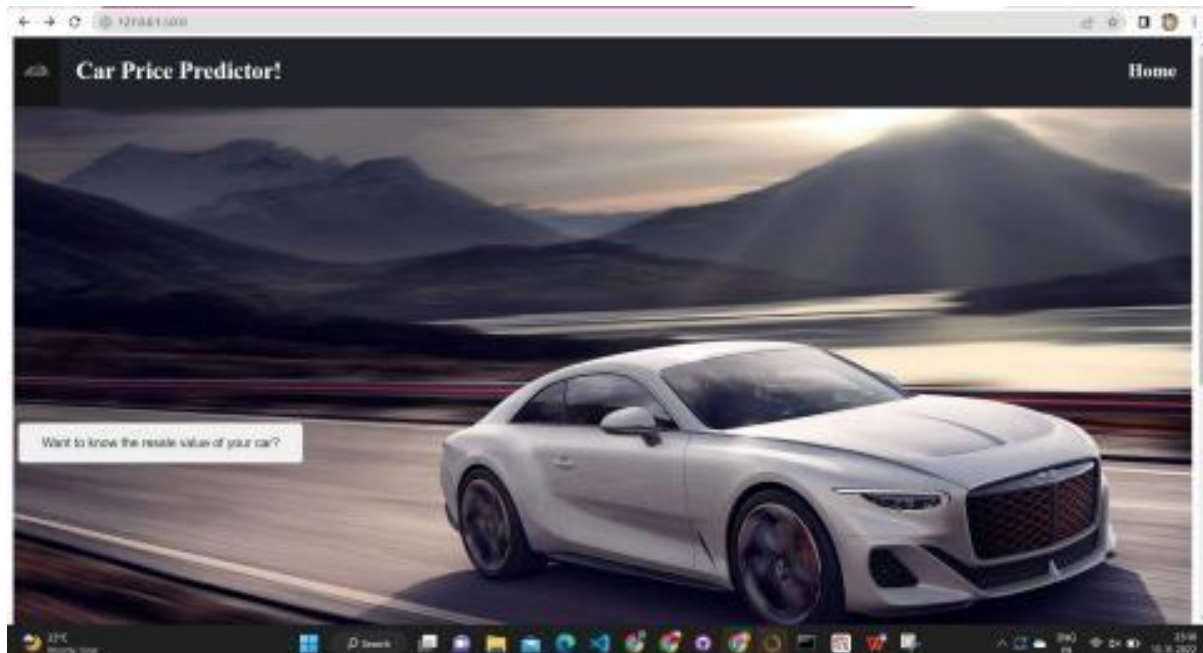
NumPy:

A collection of the multidimensional matrix that facilitates complex mathematical operations. NumPy can be used to execute operations on arrays that are related to mathematics, such as algebraic, statistical, and trigonometric patterns. The image is transformed into a matrix. The Convolutional Neural Network is utilized to understand and analyse the image in its matrix form. The image's annotations then adopted a NumPy array style. Finally, the dataset contains the precise labels for each image. On this, SciPy was also developed. It provides more noteworthy execution that utilizes NumPy arrays and is required for various logical and engineering tasks.

CHAPTER 5

5. OUTPUT:

Home page



Prediction form page:

The screenshot shows a web browser window with the URL <http://127.0.0.1:5000/predict>. The page has a large grey header area with the text "Prediction Form". Below the header, there are several input fields with labels: "Enter car details", "Registration year*", "Registration month*", "Power of car in PS*", "Automatic Drive?", and "Gear Box Type". The "Registration year" field contains the value "2013". The "Registration month" field contains the value "9". The "Power of car in PS" field contains the value "191". The "Automatic Drive?" field contains the value "Automatic". The "Gear Box Type" field contains the value "Manual".

The screenshot shows the same web browser window, but with more input fields visible. The "Automatic Drive?" field now contains the value "Automatic". The "Gear Box Type" field contains the value "Manual". Below these fields, there are more input fields: "Car damaged/spared?", "Model Type", "Brand of the car", "Fuel type of the car", and "Vehicle Type". The "Car damaged/spared?" field contains the value "Yes". The "Model Type" field contains the value "Tata". The "Brand of the car" field contains the value "Audi". The "Fuel type of the car" field contains the value "Diesel". The "Vehicle Type" field contains the value "Small car".

The screenshot shows the same web browser window, but with the "Predict" button highlighted. Below the button, there is a message: "The resale value predicted is 19440.349".

CHAPTER 6

6. CONCLUSION:

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. By performing ML models, we aim to get a better result or less error with max accuracy to predict the value of the used car. Initially, data cleaning is performed to remove the null values and outliers from the dataset then ML models are implemented to predict the price of cars. Next, with the help of data visualization features were explored deeply. The relation between the features is examined. From the report, it can be said that gradient regression regressor is the best model for the prediction for used car prices.