USED CAR PRICE PREDICITON USING MACHINE LEARNING IN PYTHON

Dissertation submitted in fulfillment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE AND ENGINEERING

By

MANAN PANDEY
12213157

Supervisor

Mr. Ved Prakash Chaubey

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Month…… Year ……

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation proposal entitled "USED CAR PRICE PREDICTION USING MACHINE LEARINING AND PYTHON" in partial fulfilment of the requirement for the award of Degree for Master of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Ved Prakash Choubey. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Name : Manan Pandey

R.No : 57

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech dissertation proposal entitled **"USED CAR PRICE PREDICTION USING MACHINE LEARNING AND PYTHON"** submitted by **Manan Pandey** at **Lovely Professional University, Phagwara, India** is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Mr. Ved Prakash Choubey

Date:

Counter Signed by:

1) **Concerned HOD:**
   HoD's Signature: _____

   HoD Name: _____

   Date: _____

2) **Neutral Examiners:**

   External Examiner

   Signature: _____

   Name: _____

   Affiliation: _____

   Date: _____

   Internal Examiner

   Signature: _____

   Name: _____

   Date: _____

TABLE OF CONTENTS

CONTENTS **PAGE NO.**

Checklist for Dissertation-III Supervisor

Name: _____ UID: _____ Domain: _____
Registration No: _____Name of student:_____
Title of Dissertation:

_____

- ☐ Front pages are as per the format.
- ☐ Topic on the PAC form and title page are same.
- ☐ Front page numbers are in roman and for report, it is like 1, 2, 3…….
- ☐ TOC, List of Figures, etc. are matching with the actual page numbers in the report.
- ☐ Font, Font Size, Margins, line Spacing, Alignment, etc. are as per the guidelines.
- ☐ Color prints are used for images and implementation snapshots.
- ☐ Captions and citations are provided for all the figures, tables etc. and are numbered and center aligned.
- ☐ All the equations used in the report are numbered.
- ☐ Citations are provided for all the references.
- ☐ **Objectives are clearly defined.**
- ☐ Minimum total number of pages of report is 50.
- ☐ Minimum references in report are 30.

Here by, I declare that I had verified the above mentioned points in the final dissertation report.

Signature of Supervisor with UID

ABSTRACT

Title: Used Car Price Prediction using Machine Learning in Python

In recent years, the automotive industry has witnessed a surge in the demand for accurate car price prediction models. These models play a crucial role in aiding potential car buyers and sellers in making informed decisions. This project aims to develop a machine learning-based system using Python that predicts car prices based on various features such as mileage, brand, model year, and condition.

The proposed system employs a robust dataset comprising comprehensive information about a wide range of car models and their corresponding market prices. Leveraging this data, the project utilizes advanced machine learning algorithms and python to train a predictive model capable of accurately estimating car prices.

To accomplish this, the project follows a systematic approach that involves data pre-processing, feature selection, model training, and evaluation. The dataset is cleaned and pre-processed to handle missing values, outliers, and categorical variables. Relevant features are then selected based on their impact on price prediction. Subsequently, the selected features are fed into various machine learning algorithms, such as linear regression, decision trees, and random forests, to train and compare multiple models.

The performance of each model is evaluated using various metrics, including mean squared error, mean absolute error, and R-squared score. The model with the best performance is then selected as the final car price prediction model.

The developed Python application provides an intuitive user interface, allowing users to input car details and obtain accurate price predictions. The application's simplicity and accessibility make it an invaluable tool for both car buyers and sellers, enhancing their decision-making process.

The results of this project demonstrate the effectiveness and reliability of machine learning techniques in predicting car prices. The developed model showcases the potential for further advancements in the field, enabling users to make well-informed decisions in the dynamic automotive market.

Keywords: car price prediction, machine learning, Python, predictive modelling, feature selection, data pre-processing.

CHAPTER-1 : INTRODUCTION

Used Car Price Prediction using Machine Learning in Python

Predicting the price of a used cars has been studied extensively in various researches.

Car price prediction is a somehow an interesting and popular problem.

As per information from the Agency for Statistics of BiH, the percentage of personal car usage is increased by 2.7% since 2013 and it is likely that this trend will continue, and the number of cars will increase day by day.

Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors.

Typically, the most significant ones are present price , brand and model, age, mileage etc. The fuel type used in the car as well as fuel consumption per mile highly affect the price of a car due to a frequent changes in the price of a fuel. Different features like exterior color, type of transmission, safety, air condition, etc. will also influence the car price.

The goal of this project is to develop a machine learning model that can accurately predict the price of a used car based on its features. By training the model on historical data, it can learn patterns and relationships between the car's features and its price. Once the model is trained, it can be used to predict the price of a new car based on its features.

To accomplish this, we will need to follow a step-by-step process that includes data pre-processing, feature selection, model training, and evaluation. Python provides a wide range of libraries and tools that make it easy to implement machine learning algorithms and perform data analysis tasks.

By the end of this project, we will have gained valuable experience in machine learning, data pre-processing, feature selection, model training, and evaluation. We will also have a working model that can predict the price of a used car based on its features.

CHAPTER-2 : PRESENT WORK

2.1 Problem Statement:

A model to predict the price of a used car should be developed in order to assess its value based on a variety of characteristics. Several factors affect the price of a used car, such as company, model, year, transmission, distance driven, fuel type, seller type, and owner type. As a result, it is crucial to know the used car's actual market value before purchasing it.

2.2 Objective of the Project:

The objective of the above-mentioned project on Used car price prediction is to develop a machine learning model using python that can accurately predict the prices of used cars based on their features. The model aims to analyze various factors such as the car's age, mileage, brand, model, and condition to make accurate price predictions.

By training the machine learning model on historical data, the objective is to enable it to learn patterns and relationships between the car's features and its price. The model will be evaluated based on its ability to accurately predict the prices of new cars based on their features.

The ultimate goal of this project is to provide a tool that can assist buyers, sellers, and other stakeholders in the used car market to make informed decisions by predicting the fair market value of a used car.

By achieving this objective, the project aims to contribute to the field of machine learning and predictive analytics, as well as provide practical value to individuals and businesses involved in the used car market.

Remember to cite the relevant sources using the [[SOURCE #]] notation throughout your project to support your claims and findings.

Good luck with your project, and may it bring valuable insights into the world of used car price prediction!

2.3Approach:

- Step 1: "Car_Data.csv" dataset was taken from kaggle.com and was reconfigured to reflect the important features.
  Data frames in which we loaded the dataset now include the Selling price , Present price, Kms driven, owner, age, Fuel Type, Seller type, Transmission type.

- Step 2: Categorical features were then converted to numerical values.

  We remove multiple columns of the dataset using the get_dummies function as some columns contain the same information because the original column could assume a binary value, The "CNG" fuel type, the "Automatic" transmission type, and the "Dealer" seller type were removed.

- Step 3: We created a heat map using Pearson correlation to illustrate how the features are related.

- Step 4: Using ExtraTreesRegressor the feature importances were obtained. And Present_Price turned out to be the feature with most importance. We used Random Forest Regression to solve the problem and created a model based on this.
- Step 5: By using randomizedsearchCV, we performed hyper parameter tuning (n_estimators, max_features, max_depth, min_samples_split, min_samples_leaves). Due to its speed, RandomizedSearchCV is better than GridSearchCV.

- Step 6: The data has been split into training data (80%) and test data (20%).

- Step 7: After getting the best parameters from the RandomizedSearchCV we train the model using the Random Forest Regression . In the random forest regressor, the decision tree will scale the input, so we do not have to scale the values. The graph of predictions is plotted using displot/ distplot in which we observe that it resembles a normal distribution with mean 0.

- Step 8 : The file is put in a pickle file. Also the performances of the model are computed.

- Step 9 : Lastly, the final output is displayed on the HTML page. The model is demonstrated by entering features in an HTML file and predicting the price by using a Python file from the set of values given as features.

CHAPTER-3 : RESULTS AND DISCUSSION

Car Price Prediction

In [1]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv('car data.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [6]:
```
df.shape
```

Out[6]:
```
(301, 9)
```

In [7]:
```
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
[0 1 3]
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Car_Name         0
Year             0
```

jupyter **carprice** Last Checkpoint: 14 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [10]:
```
final_dataset = df[['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner']]
```

In [11]:
```
final_dataset.head()
```

Out[11]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [12]:
```
final_dataset['Current_Year'] = 2021
```

In [13]:
```
final_dataset.head()
```

In [13]:
```
final_dataset.head()
```

Out[13]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021

In [14]:
```
final_dataset['Age'] = final_dataset['Current_Year']-final_dataset['Year']
```

In [15]:
```
final_dataset.head()
```

Out[15]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	Age
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

In [16]:
```
final_dataset.drop(['Year'],axis=1,inplace=True)
#drops the column labelled as "year" and doesnt return a copy as inplace = true. and axis = 1 represents columns
```

In [17]:
```
final_dataset.drop(['Current_Year'],axis=1,inplace=True)
#drops the column labelled as "Current_year" and doesnt return a copy as inplace = true. and axis = 1 represents col
```

In [18]:
```
final_dataset.head()
```

Out[18]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age

localhost

Home Page - Select or create a notebook | carprice - Jupyter Notebook

jupyter **carprice** Last Checkpoint: 14 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [18]:
```
final_dataset.head()
```

Out[18]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	7
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	8
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	4
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	10
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	7

In [19]:
```
print(df['Fuel_Type'].unique())
```

```
['Petrol' 'Diesel' 'CNG']
```

In [20]:
```
final_dataset=pd.get_dummies(final_dataset,drop_first=True)
#removes multiple columns of the dataset as some column contain the same
#information because the original column could assume a binary value.
```

In [21]:
```
final_dataset.head()
```

Out[21]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1
1	4.75	9.54	43000	0	8	1	0	0	1
2	7.25	9.85	6900	0	4	0	1	0	1
3	2.85	4.15	5200	0	10	0	1	0	1
4	4.60	6.87	42450	0	7	1	0	0	1

In [22]:
```
final_dataset
```

Out[22]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1

localhost

Home Page - Select or create a notebook | carprice - Jupyter Notebook

jupyter **carprice** Last Checkpoint: 14 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [22]:
```
final_dataset
```

Out[22]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1
1	4.75	9.54	43000	0	8	1	0	0	1
2	7.25	9.85	6900	0	4	0	1	0	1
3	2.85	4.15	5200	0	10	0	1	0	1
4	4.60	6.87	42450	0	7	1	0	0	1
...	...	...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	5	1	0	0	1
297	4.00	5.90	60000	0	6	0	1	0	1
298	3.35	11.00	87934	0	12	0	1	0	1
299	11.50	12.50	9000	0	4	1	0	0	1
300	5.30	5.90	5464	0	5	0	1	0	1

301 rows × 9 columns

In [23]:
```
final_dataset.corr(method ='pearson')
#to find the pairwise correlation of all columns in the dataframe
```

Out[23]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_I
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.:
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.:
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.:
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.(
Age	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.(
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.(
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.(

localhost

Home Page - Select or create a notebook | carprice - Jupyter Notebook

jupyter **carprice** Last Checkpoint: 14 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[23]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_M
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.3
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.3
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.
Age	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.000000	0.
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.063240	1.

In [24]:
```
import matplotlib.pyplot as plt
%matplotlib inline
#to display the plot directly below the code cell.
```

In [25]:
```
corrmat = df.corr(method='pearson')
```

In [26]:
```
corrmat.index
```

Out[26]:
```
Index(['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner'], dtype='object')
```

In [27]:
```
import seaborn as sns
#get correlations of each features in dataset
corrmat = df.corr(method='pearson')
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map of the corerlation of the features
g=sns.heatmap(df[top_corr_features].corr(method='pearson'),annot=True,cmap="RdYlGn")
#annot = true will write the data in each cell.
#cmap sets the color of the maps
```

Safari File Edit View History Bookmarks Window Help Wed 8 Nov 12:11 AM

localhost

Home Page - Select or create a notebook | carprice - Jupyter Notebook | iyashk/Car-Price-Prediction: A Machine Learning Project that uses Random For...

jupyter **carprice** Last Checkpoint: 28 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [29]:
```
final_dataset
```

Out[29]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1
1	4.75	9.54	43000	0	8	1	0	0	1
2	7.25	9.85	6900	0	4	0	1	0	1
3	2.85	4.15	5200	0	10	0	1	0	1
4	4.60	6.87	42450	0	7	1	0	0	1
...	...	...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	5	1	0	0	1
297	4.00	5.90	60000	0	6	0	1	0	1
298	3.35	11.00	87934	0	12	0	1	0	1
299	11.50	12.50	9000	0	4	1	0	0	1
300	5.30	5.90	5464	0	5	0	1	0	1

301 rows × 9 columns

In [30]:
```
final_dataset.iloc[:,0]
```

Out[30]:
```
0      3.35
1      4.75
2      7.25
3      2.85
4      4.60
      ...
296    9.50
```

jupyter carprice Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [30]:
```
final_dataset.iloc[:,0]
```

Out[30]:
```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

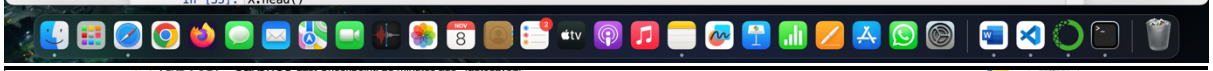
In [31]:
```
X= final_dataset.iloc[:,1:]
#slicing the dataset and reomoving the selling price for training the model
Y = final_dataset.iloc[:,0]
#storing the selling price for checking..as this is the value to be predicted
```

In [32]:
```
final_dataset.head()
```

Out[32]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1
1	4.75	9.54	43000	0	8	1	0	0	1
2	7.25	9.85	6900	0	4	0	1	0	1
3	2.85	4.15	5200	0	10	0	1	0	1
4	4.60	6.87	42450	0	7	1	0	0	1

In [33]:
```
X.head()
```



File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [33]:
```
X.head()
```

Out[33]:

	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	5.59	27000	0	7	0	1	0	1
1	9.54	43000	0	8	1	0	0	1
2	9.85	6900	0	4	0	1	0	1
3	4.15	5200	0	10	0	1	0	1
4	6.87	42450	0	7	1	0	0	1

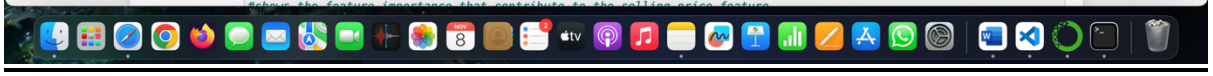
In [34]:
```
Y.head()
```

Out[34]:
```
0    3.35
1    4.75
2    7.25
3    2.85
4    4.60
Name: Selling_Price, dtype: float64
```

In [35]:
```
from sklearn.ensemble import ExtraTreesRegressor
#in ensemble predictions of several base estimators are built in with a given learning algorithm.
#we used ExtraTreesRegressor
model = ExtraTreesRegressor()
#This class implements a meta estimator that fits a number of randomized decision trees
#on various sub-samples of the dataset and uses averaging to improve the
#predictive accuracy and control over-fitting.
model.fit(X,Y)
```

Out[35]:
```
ExtraTreesRegressor()
```

In [36]:
```
print(model.feature_importances_)
#shows the features importances that contributes to the selling price feature
```



jupyter carprice Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

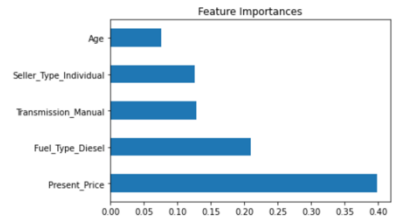
Trusted | Python 3 (ipykernel)

Run Code

In [36]:
```
print(model.feature_importances_)
#shows the feature importance that contribute to the selling price feature
```

```
[0.39889346 0.03983609 0.0009349  0.07578666 0.20973752 0.02023177
 0.12591882 0.12866078]
```

In [37]:
```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```



In [38]:
```
from sklearn.model_selection import train_test_split #class to divide the data into train and validation set

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
#we divide the data into 2 parts :- 80% train and 20% test data
# and random_state is used to guarantee that same sequence of
#random numbers are generated each time you run the code.
#And unless there is some other randomness present in the process,
```

jupyter carprice Last Checkpoint: 37 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [38]:
```
from sklearn.model_selection import train_test_split #class to divide the data into train and validation set

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
#we divide the data into 2 parts :- 80% train and 20% test data
# and random_state is used to guarantee that same sequence of
#random numbers are generated each time you run the code.
#And unless there is some other randomness present in the process,
#the results produced will be same as always.
```

In [39]:
```
X_train.shape
```

Out[39]: (240, 8)

In [40]:
```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
```

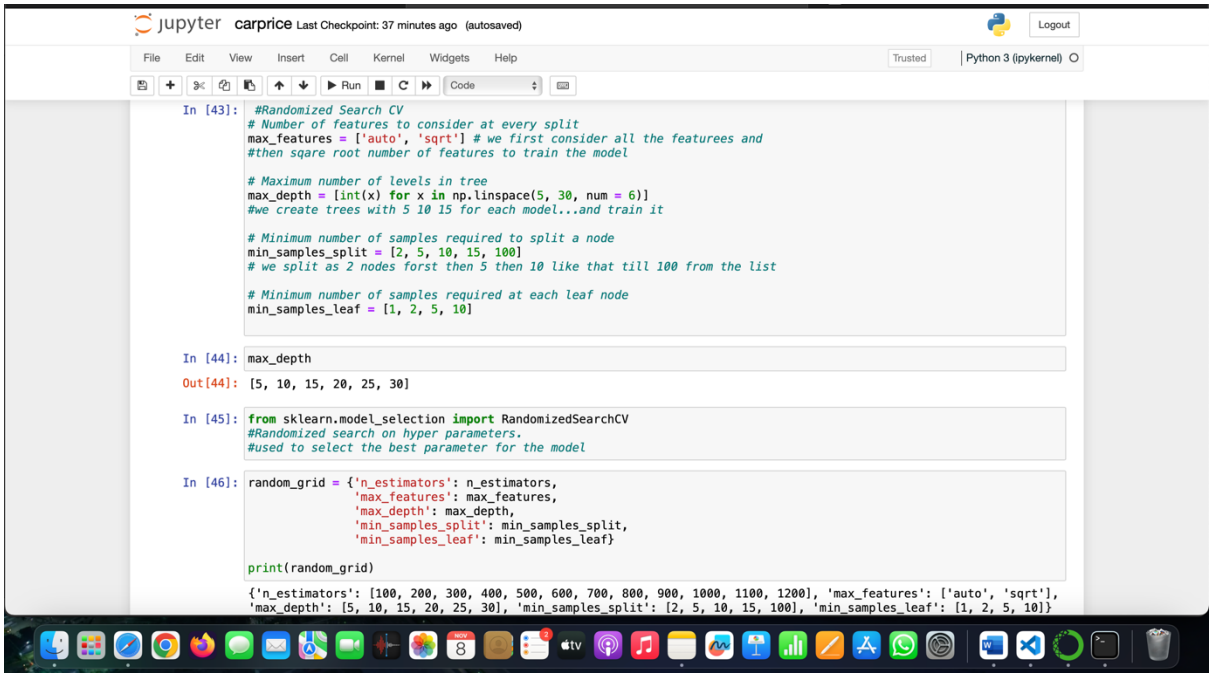
In [41]:
```
#A random forest regressor is an estimator that fits a number of classifying decision trees
#on various sub-samples of the dataset and uses averaging to improve the prediction accuracy
#and slos control over-fitting.
```

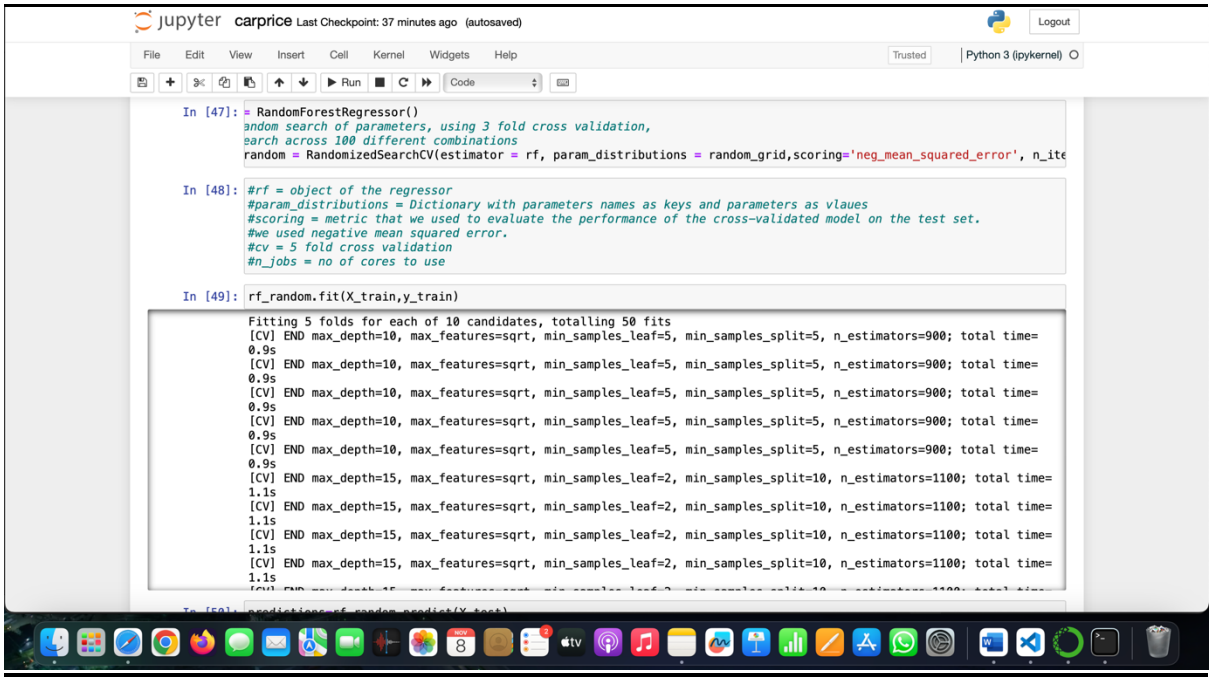
In [42]:
```
import numpy as np
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
#n_estimators is a parameter of the random forest regressor which is used to control no of trees in the forest
#so we use 100 200 ....1200 trees for the model
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

In [43]:
```
 #Randomized Search CV
# Number of features to consider at every split
max_features = ['auto', 'sqrt'] # we first consider all the featurees and
#then sqare root number of features to train the model

# Maximum number of levels in tree
```


jupyter carprice Last Checkpoint: 37 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [43]:
#Randomized Search CV
# Number of features to consider at every split
max_features = ['auto', 'sqrt'] # we first consider all the featurees and
#then sqare root number of features to train the model
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
#we create trees with 5 10 15 for each model...and train it
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# we split as 2 nodes forst then 5 then 10 like that till 100 from the list
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
In [44]:
max_depth
Out[44]:
[5, 10, 15, 20, 25, 30]
In [45]:
from sklearn.model_selection import RandomizedSearchCV
#Randomized search on hyper parameters.
#used to select the best parameter for the model
In [46]:
random_grid = {'n_estimators': n_estimators,
'max_features': max_features,
'max_depth': max_depth,
'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf}
print(random_grid)
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqrt'],
'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}


jupyter carprice Last Checkpoint: 37 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [47]:
= RandomForestRegressor()
andom search of parameters, using 3 fold cross validation,
earch across 100 different combinations
random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_ite
In [48]:
#rf = object of the regressor
#param_distributions = Dictionary with parameters names as keys and parameters as vlaues
#scoring = metric that we used to evaluate the performance of the cross-validated model on the test set.
#we used negative mean squared error.
#cv = 5 fold cross validation
#n_jobs = no of cores to use
In [49]:
rf_random.fit(X_train,y_train)
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
0.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
0.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
0.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
0.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=
1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=
1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=
1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=
1.1s

jupyter **carprice** Last Checkpoint: 37 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

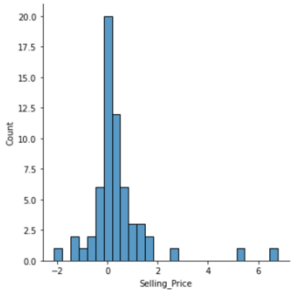
Run Code

In [50]:
```
predictions=rf_random.predict(X_test)
```

In [51]:
```
predictions1=rf_random.predict(X_train)
```

In [52]:
```
sns.displot(y_test-predictions)
```

Out[52]: <seaborn.axisgrid.FacetGrid at 0x7fbe50fcc760>



In [53]:
```
sns.displot(y_train-predictions1)
```

Out[53]: <seaborn.axisgrid.FacetGrid at 0x7fbe62aff190>

jupyter **carprice** Last Checkpoint: 37 minutes ago (autosaved)

Logout

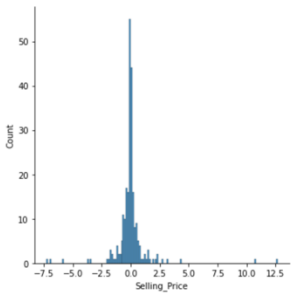
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [53]:
```
sns.displot(y_train-predictions1)
```

Out[53]: <seaborn.axisgrid.FacetGrid at 0x7fbe62aff190>



In [54]:
```
#form the plot we can see that almost all the values predicted are correct by comparing the diff between the y_test
#predicted value and we see from the normal distribution of the graph that not much of the samples have varied
#much from the actual value
```

In [55]:
```
plt.scatter(y_test,predictions)
```

Out[55]: <matplotlib.collections.PathCollection at 0x7fbe5a665850>

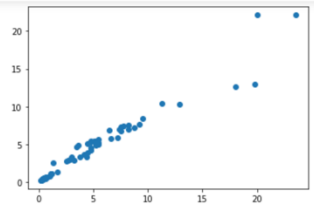
jupyter carprice Last Checkpoint: 38 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

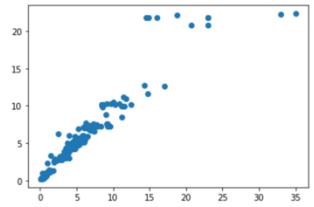
Trusted | Python 3 (ipykernel)

Run Code



In [56]:
```
plt.scatter(y_train,predictions1)
```

Out[56]: <matplotlib.collections.PathCollection at 0x7fbe510fbfd0>



In [57]:
```
#even from this points plot we can see that almost all the points are along the
```

jupyter carprice Last Checkpoint: 38 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [57]:
```
#even from this points plot we can see that almost all the points are along the
#line y=x which says that the predicted and the y_test values are smae
```

In [58]:
```
import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
#we use a pickle file to store the data in a byte stream format
```

In [59]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test,predictions)
rmse = np.sqrt(mse)
print("RMSE : {:.2f}".format(rmse))
```

```
RMSE : 1.34
```

In [60]:
```
from sklearn.metrics import r2_score
r = r2_score(y_test, predictions)
print("R2 score : {}" . format(r))
```

```
R2 score : 0.9288666495406369
```

In [61]:
```
df_check = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
df_check = df_check.head(25)
#round(df_check,2)
df_check.plot(kind='bar',figsize=(10,5))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.title('Performance of Random Forest')
plt.ylabel('Price')
plt.show()
```

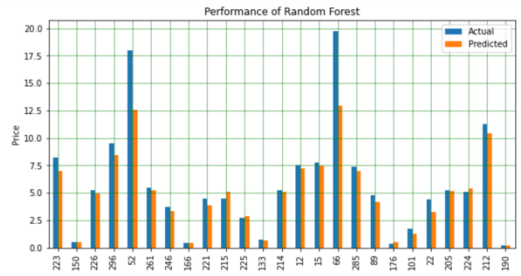
Performance of Random Forest

jupyter **carprice** Last Checkpoint: 40 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3 (ipykernel)

Run | Code



In [62]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
mse_predict = round(mean_squared_error(y_test,predictions),4)
mae_predict = round(mean_absolute_error(y_test,predictions),4)
print ('MSE is:'+str(mse_predict))
print ('MAE is:'+str(mae_predict))
```

```
MSE is:1.7981
MAE is:0.7026
```

In [63]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [64]:
```
cutoff = 1.8                            # decide on a cutoff limit
y_pred_classes = np.zeros_like(predictions)
```

jupyter **carprice** Last Checkpoint: 40 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3 (ipykernel)

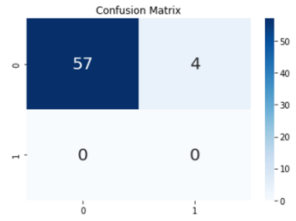
Run | Code

In [64]:
```
cutoff = 1.8                            # decide on a cutoff limit
y_pred_classes = np.zeros_like(predictions)
y_test_classes = np.zeros_like(predictions)     # initialise a matrix full with zeros
y_pred_classes[abs(y_test - predictions) > cutoff] = 1
```

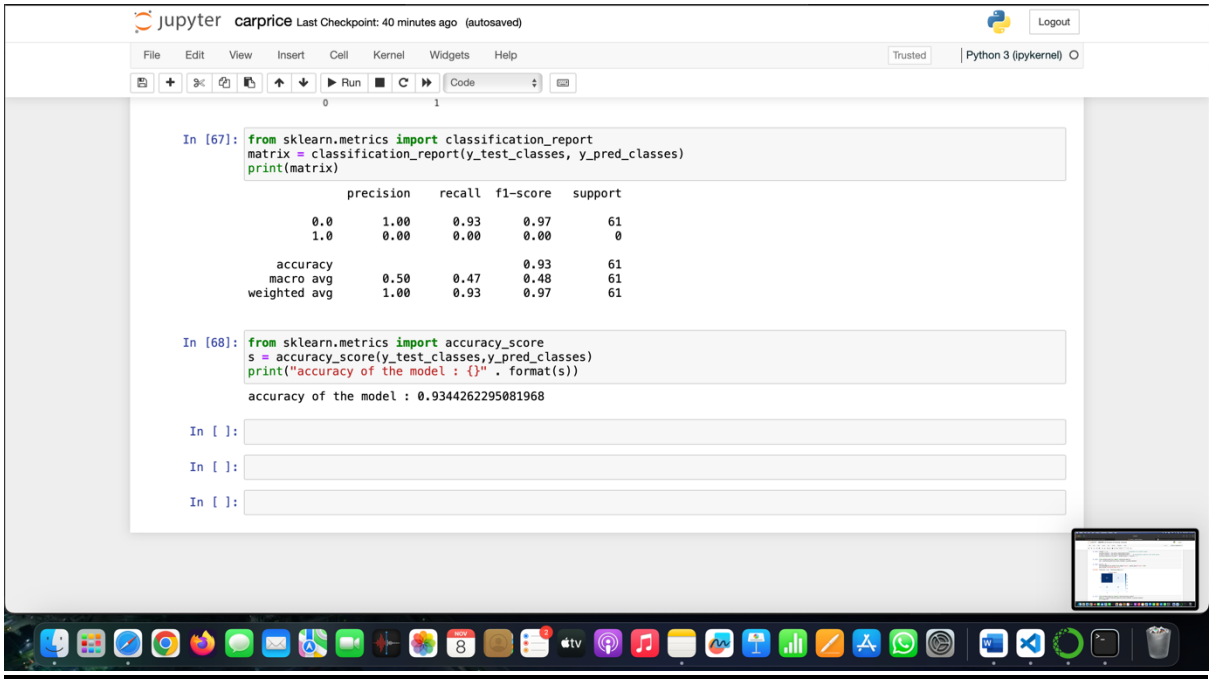
In [65]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test_classes, y_pred_classes)
```

In [66]:
```
df_cm = cm
sns.heatmap(df_cm,annot=True,cmap="Blues" ,annot_kws={"size": 20})
plt.title('Confusion Matrix')
```

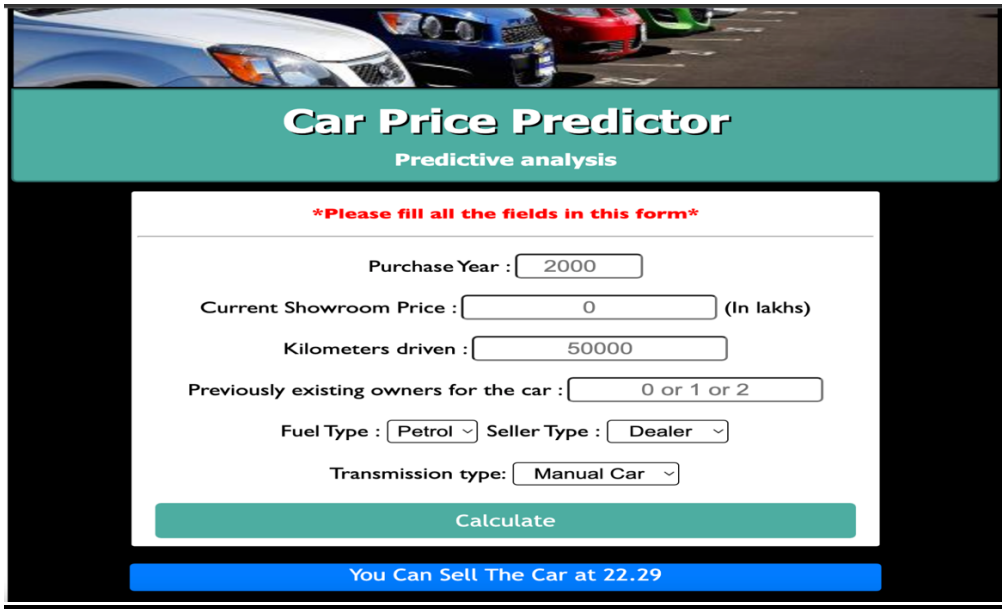
Out[66]: Text(0.5, 1.0, 'Confusion Matrix')



In [67]:
```
from sklearn.metrics import classification_report
matrix = classification_report(y_test_classes, y_pred_classes)
print(matrix)
```


jupyter carprice Last Checkpoint: 40 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [67]:
from sklearn.metrics import classification_report
matrix = classification_report(y_test_classes, y_pred_classes)
print(matrix)
precision recall f1-score support
0.0 1.00 0.93 0.97 61
1.0 0.00 0.00 0.00 0
accuracy 0.93 61
macro avg 0.50 0.47 0.48 61
weighted avg 1.00 0.93 0.97 61
In [68]:
from sklearn.metrics import accuracy_score
s = accuracy_score(y_test_classes,y_pred_classes)
print("accuracy of the model : {}" . format(s))
accuracy of the model : 0.9344262295081968
In [ ]:
In [ ]:
In [ ]:

HTML PAGE:


Car Price Predictor
Predictive analysis
Please fill all the fields in this form
Purchase Year : 2000
Current Showroom Price : 0 (In lakhs)
Kilometers driven : 50000
Previously existing owners for the car : 0 or 1 or 2
Fuel Type : Petrol
Seller Type : Dealer
Transmission type: Manual Car
Calculate
You Can Sell The Car at 22.29

CHAPTER4: CONCLUSION AND FUTURE SCOPE

4.1 Conclusion :

In this ever-evolving world of technology, the integration of machine learning and data analysis has proven to be a game-changer in various industries. The car price prediction project presented here is a testament to the power and potential of this cutting-edge technology.

By harnessing the vast amount of data available in the automotive market, this project has successfully developed a robust and accurate car price prediction system. Through the use of advanced algorithms and comprehensive datasets, the machine learning model has been trained to identify patterns and correlations, enabling it to provide reliable estimates of car prices.

The significance of this project extends beyond the realm of convenience for buyers and sellers. It serves as a stepping stone towards greater market transparency and fairness. With the ability to accurately predict car prices, potential buyers can make well-informed decisions, ensuring that they are paying a fair price for their desired vehicle. Similarly, sellers can optimize their pricing strategies, maximizing their chances of a successful sale.

The Python application developed as part of this project showcases the user-friendly interface that empowers individuals with the ability to obtain instant price predictions. This accessibility and ease of use further enhance the project's impact, making it a valuable tool for all participants in the automotive market.

As we look to the future, the possibilities for further refinement and expansion of this car price prediction system are endless. By incorporating additional features, such as real-time market data, user feedback, and even sentiment analysis, the accuracy and reliability of the model can be enhanced even further.

The journey of this project has been one of exploration, innovation, and collaboration. It highlights the immense potential of machine learning in revolutionizing the way we navigate the automotive landscape. With each prediction made, this project takes a step towards a future where data-driven decisions are the norm, reshaping the way we buy and sell cars.

In conclusion, the car price prediction project presented here is a testament to the power of machine learning and python in transforming industries. By accurately estimating car prices, it empowers buyers and sellers alike, fostering transparency, fairness, and informed decision-making. As technology continues to advance, this project sets the stage for further advancements in the automotive market, paving the way for a future where data-driven intelligence reigns supreme.

4.2 Future Scope :

The project " *Used Car Price Prediction using Machine Learning in Python* " has a promising future scope.
Here are some potential areas of development and application:

1. Improved Prediction Models : The project can explore the use of advanced machine learning algorithms and techniques to enhance the accuracy of used car price prediction. For example, ensemble methods like Random Forest Regression and deep learning models such as neural networks can be employed to achieve better results.

2. Feature Engineering : Further research can be conducted to identify and incorporate additional relevant features that can contribute to more accurate price predictions. This may involve analysing factors such as car condition, mileage, location, and historical sales data.

3. Data Collection and Integration : Expanding the dataset used for training the prediction models can lead to more robust and reliable results. This can involve scraping data from various online platforms, integrating multiple data sources, and ensuring data quality and consistency.

4. Real-Time Price Updates : Developing a system that continuously updates the predicted prices based on real-time market trends and fluctuations can be valuable for both buyers and sellers. This can involve integrating APIs or web scraping techniques to gather up-to-date information on car prices.

5. User-Friendly Interface : Creating a user-friendly interface or application that allows users to input car details and obtain accurate price predictions can enhance the usability and accessibility of the project. This can involve developing a web or mobile application using frameworks like Flask or Django.

6. Integration with Online Marketplaces : Integrating the price prediction model with popular online car marketplaces can provide users with instant price estimates for their listed vehicles. This can enhance the user experience and attract more users to the platform.

7. Expansion to Other Vehicle Types : While the project focuses on used car price prediction, the techniques and models developed can be extended to other types of vehicles, such as motorcycles, trucks, or boats. This can broaden the scope of the project and cater to a wider audience.

8. Integration with Car Valuation Services : Collaborating with established car valuation services can provide access to industry expertise and data, further improving the accuracy of the price prediction models.

9. Market Analysis and Insights : Analysing the predicted prices and market trends can provide valuable insights into the used car market. This information can be used by car dealers, buyers, and sellers to make informed decisions.

It is important to note that the future scope of the project will depend on the availability of data, advancements in machine learning techniques, and the level of effort and resources dedicated to its development.

Overall, the project has significant potential for further development and application in the field of automotive sales and pricing.