18CS3166S – MACHINE LEARNING

PROJECT BASED REPORT

ON

Automobile Price Prediction using Regression Models

submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

B.Purvaja Durga 180030593

(Batch Number : 10)

Under the Esteemed Guidance of

Dr.T.SAJANA

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K L E F

Green Fields, Vaddeswaram, Guntur District – 522 502

(2020-2021)

CERTIFICATE

This is certify that the project based report entitled “Automobile Price Prediction using Regression models” is a bonafide work done and submitted by B.PURVAJA DURGA (180030593) in partial fulfillment of there requirements for the award of the degree of BACHELOR OF TECHNOLOGY in Department of Computer Science Engineering, KLEF Guntur District during the academic year 2020-2021.

FACULTY INCHARGE **HEAD OF THE DEPARTMENT**

Dr.T.SANJANA **HARIKIRAN VEGE**

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our sincere thanks to all those who has assisted us in one way or the other for the completion of my project.

Our greatest appreciation to my guide **Dr.T.SAJANA**, Assistant Professor, Department of Computer Science and Engineering which cannot be express in words for her tremendous support, encouragement and guidance for this project.

We express our gratitude to **Dr. Hari Kiran Vege,** Head of the Department for Computer Science and Engineering for providing us with adequate facilities, ways and means by which we are able to complete this project.

We thank all the members of teaching and non-teaching staff members, and also who have assisted me directly or indirectly for successful completion of this project.

Finally, I sincerely thank my friends and classmates for their kind help and co- operation during our work.

B.PURVAJA DURGA

180030593

INDEX

INTRODUCTION

Approximately 40 million vehicles are sold each year. Effective pricing strategies can help any company to efficiently sell its products in a competitive market and making profit.

In the automotive sector, pricing analytics play an essential role for both companies and individuals to assess the market price of a vehicle before putting it on sale or buying it.

There are two main goals I want to achieve with this Project. First, to estimate the price of used cars by taking into account a set of features, based on historical data. Second, to get a better understanding on the most relevant features that help determine the price of a used vehicle.

The focus of this project is developing machine learning models that can accurately predict the price of a car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in the United States.

Deciding whether a car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately[2-3]. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

About Dataset:

The data that will be used for this project is accessible at *Kaggle*

Get the data: The first thing we need in machine learning is data. We'll use the sample dataset, Automobile price data (Raw). This dataset includes entries for various individual automobiles, including information such as make, model, technical specifications, and price.

Prepare the data: A dataset usually requires some pre processing before it can be analyzed. You might have noticed the missing values present in the columns of various rows. These missing values need to be cleaned so the model can analyze the data correctly.

In machine learning, features are individual measurable properties of something you're interested in. In our dataset, each row represents one automobile, and each column is a feature of that automobile.

Let's build a model that uses a subset of the features in our dataset. You can come back later and select different features, run the experiment again, and see if you get better results.

METHODOLOGY

Problem Statement :

Automobile price prediction using Regression models

Dataset Description :

Before pre-processing the data we must take a look to how the dataset shows up. In particular we carry out an analysis on the price attribute: describing it allows us to appreciate some information’s such as min and max values and standard deviation.

PRE-PROCESSING :

- Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.
- Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Handling Null Values :

First of all, we need to check whether we have null values in our dataset or not. We can do that using the isnull() method.

- df.isnull()

  Returns a boolean matrix, if the value is NaN then True otherwise False
- df.isnull().sum()

  # Returns the column names along with the number of NaN values in that particular column
- In this data set there are no null values
-

Handling Categorical Variables :

Handling categorical variables is another integral aspect of Machine Learning. Categorical variables are basically the variables that are discrete and not continuous

Using Label Encoder :

Label Encoder can be used to normalize labels. It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels. Transform labels back to original encoding.

Exploration Data Analysis (EDA) :

The numerical features play a big role in this Regression model, so it is important to understand well how are they distributed in the Database.

```
In [4]:   1 df.info()
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB

In [5]:   1 df.columns

Out[5]: Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')

In [6]:   1 df.isnull().sum()
          2 # there are no null values

Out[6]: car_ID            0
        symboling         0
        CarName           0
        fueltype          0
        aspiration        0
        doornumber        0
```

Feature Description :

A feature is a measurable property of the object you're trying to analyze. In datasets, features appear as columns:

Load Dataset

In [2]:
```
df = pd.read_csv("E:\ML\dataset\project\dataset\price.csv")
df.head()
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19

5 rows × 26 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

The image above contains a snippet of data from a cars dataset with information about different cars . Each feature, or column, represents a measurable piece of data that can be used for analysis: CarName, fueltype, carbody, wheelbase…. and so on. Features are also sometimes referred to as “variables” or “attributes.” Depending on what you’re trying to analyze, the features you include in your dataset can vary widely.

CarName : Which describes the car name of different cars.

Fueltype : It gives whetherfuel is gas or diesel

Carbody : One important factor that impacts this decision is the type of car body.car body styles or the type/form of vehicle design.

Wheelbase : A car's wheelbase is the distance between the centres of the front and rear wheels.

Carlength : which describes the length of different cars.

Carwidth : which describes the width of different cars.

Carheight : which describes the height of different cars.

Enginesize : Car engine sizes are normally specified in litres, which is rounded up to the nearest tenth of a litre.

Boreratio : Bore-Stroke Ratio is the ratio between the dimensions of the engine cylinder bore diameter to its piston stroke-length.

Stroke : A stroke refers to the full travel of the piston along the cylinder, in either direction.

Horsepower : Horsepower refers to the power an engine produces.

Peakrpm : RPM stands for revolutions per minute, and it's used as a measure of how fast any machine is operating at a given time.

ML technique algorithm :

In this project we are using Regression Models .

The various Regression Models :

- linear regression model
- Random Forest Regression
- Decision tree regression
- Ridge Regression

Linear regression model :

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

Random Forest Regression :

Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to overfitting. Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.

Decision tree Regression :

Decision tree is an algorithm that uses a tree like graph or model of decisions and their possible outcomes to predict the final decision, this algorithm uses conditional control statement. A Decision tree is an algorithm for approaching discrete-valued target functions, in which decision tree is denoted by a learned function. For inductive learning these types of algorithms are very famous and have been successfully applied to abroad range of tasks. We give label to a new transaction that is whether it is legit or fraud for which class label is unknown and then transaction value is tested against the decision tree, and after that from root node to output/class label for that transaction a path is traced.

Ridge Regression :

Ridge Regression is a technique used when the data suffers from multicollinearity (independent variables are highly correlated). In multicollinearity, even though the least squares estimates (OLS) are unbiased, their variances are large which deviates the observed value far from the true

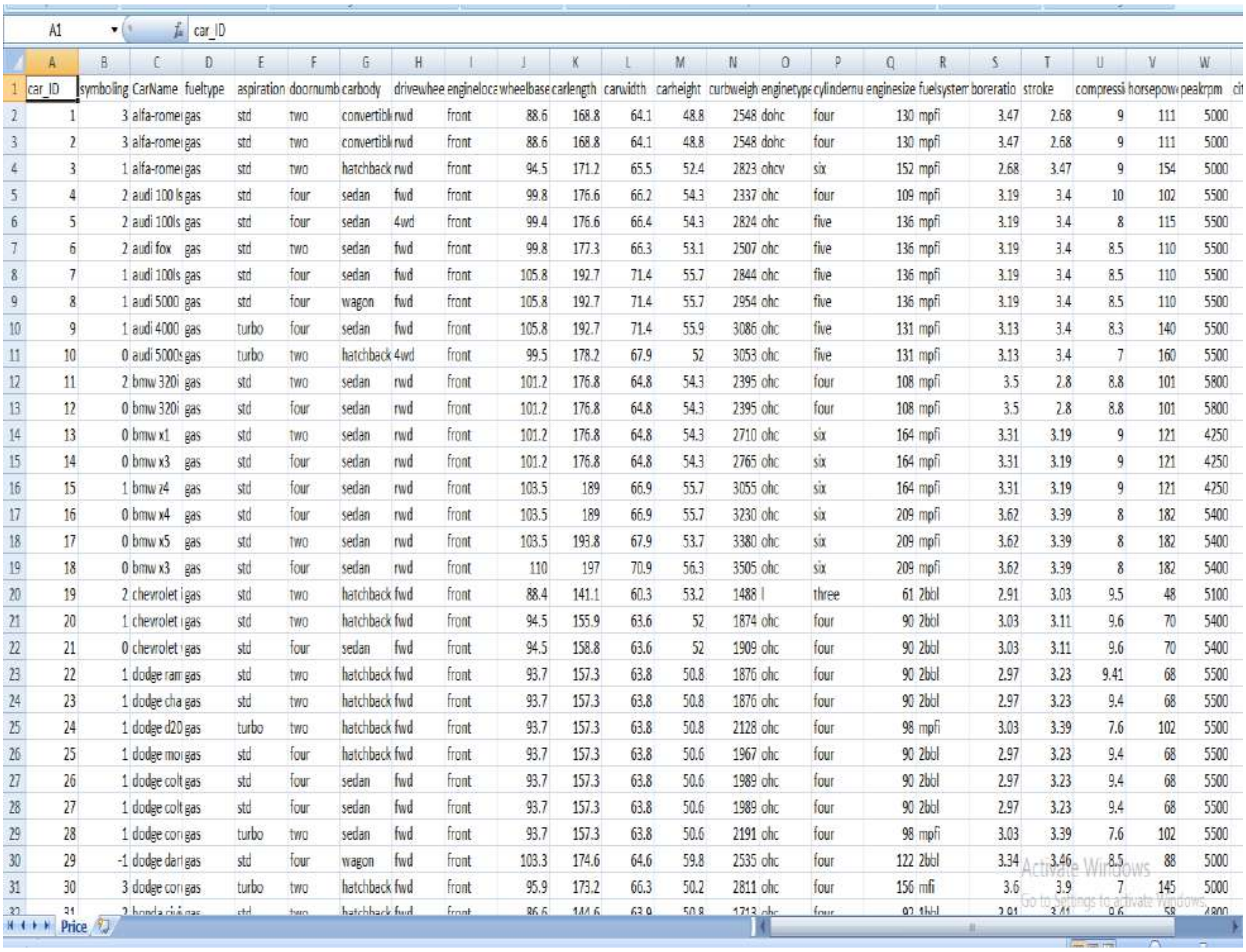
value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

RESULTS AND DISCUSSION

STEPS:

- Importing Libraries, Loading the dataset and Manipulating the data
- Splitting the dataset into Input(Features) and Output(Target)
- Data Preprocessing (Handling Missing Values)
- Data Visualization
- Splitting into training and testing data
- Build Linear Regression Model
- Build Random Forest Model
- Build Decision Tree Model
- Build Ridge Regression Model

Dataset of our project :



car_ID	symboling	CarName	fueltype	aspiration	doornumb	carbody	drivewhee	engineloca	wheelbase	carlength	carwidth	carheight	curbweigh	enginetype	cylindernu	enginesize	fuelsystem	boreratio	stroke	compressi	horsepow	peakrpm
1	3	alfa-romei	gas	std	two	convertibl	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9	111	5000
2	3	alfa-romei	gas	std	two	convertibl	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9	111	5000
3	1	alfa-romei	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	9	154	5000
4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.4	10	102	5500
5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.4	8	115	5500
6	2	audi fox	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500
7	1	audi 100ls	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7	2844	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500
8	1	audi 5000	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	55.7	2954	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500
9	1	audi 4000	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9	3086	ohc	five	131	mpfi	3.13	3.4	8.3	140	5500
10	0	audi 5000s	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9	52	3053	ohc	five	131	mpfi	3.13	3.4	7	160	5500
11	2	bmw 320i	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	8.8	101	5800
12	0	bmw 320i	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	8.8	101	5800
13	0	bmw x1	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2710	ohc	six	164	mpfi	3.31	3.19	9	121	4250
14	0	bmw x3	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2765	ohc	six	164	mpfi	3.31	3.19	9	121	4250
15	1	bmw z4	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3055	ohc	six	164	mpfi	3.31	3.19	9	121	4250
16	0	bmw x4	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3230	ohc	six	209	mpfi	3.62	3.39	8	182	5400
17	0	bmw x5	gas	std	two	sedan	rwd	front	103.5	193.8	67.9	53.7	3380	ohc	six	209	mpfi	3.62	3.39	8	182	5400
18	0	bmw x3	gas	std	four	sedan	rwd	front	110	197	70.9	56.3	3505	ohc	six	209	mpfi	3.62	3.39	8	182	5400
19	2	chevrolet i	gas	std	two	hatchback	fwd	front	88.4	141.1	60.3	53.2	1488	l	three	61	2bbl	2.91	3.03	9.5	48	5100
20	1	chevrolet i	gas	std	two	hatchback	fwd	front	94.5	155.9	63.6	52	1874	ohc	four	90	2bbl	3.03	3.11	9.6	70	5400
21	0	chevrolet	gas	std	four	sedan	fwd	front	94.5	158.8	63.6	52	1909	ohc	four	90	2bbl	3.03	3.11	9.6	70	5400
22	1	dodge ram	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8	50.8	1876	ohc	four	90	2bbl	2.97	3.23	9.41	68	5500
23	1	dodge cha	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8	50.8	1876	ohc	four	90	2bbl	2.97	3.23	9.4	68	5500
24	1	dodge d20	gas	turbo	two	hatchback	fwd	front	93.7	157.3	63.8	50.8	2128	ohc	four	98	mpfi	3.03	3.39	7.6	102	5500
25	1	dodge mo	gas	std	four	hatchback	fwd	front	93.7	157.3	63.8	50.6	1967	ohc	four	90	2bbl	2.97	3.23	9.4	68	5500
26	1	dodge colt	gas	std	four	sedan	fwd	front	93.7	157.3	63.8	50.6	1989	ohc	four	90	2bbl	2.97	3.23	9.4	68	5500
27	1	dodge colt	gas	std	four	sedan	fwd	front	93.7	157.3	63.8	50.6	1989	ohc	four	90	2bbl	2.97	3.23	9.4	68	5500
28	1	dodge cor	gas	turbo	two	sedan	fwd	front	93.7	157.3	63.8	50.6	2191	ohc	four	98	mpfi	3.03	3.39	7.6	102	5500
29	-1	dodge dar	gas	std	four	wagon	fwd	front	103.3	174.6	64.6	59.8	2535	ohc	four	122	2bbl	3.34	3.46	8.5	88	5000
30	3	dodge cor	gas	turbo	two	hatchback	fwd	front	95.9	173.2	66.3	50.2	2811	ohc	four	156	mfi	3.6	3.9	7	145	5000

OUTPUTS :

Automobile Price Prediction using Regression models

Steps:

```
1. Load all Libraries
2. Load the Dataset
3. Split the dataset
4. Fit the model
5. Make Predictions
```

Import Libraries

In [59]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn as sns
from matplotlib.pyplot import xticks
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error
import numpy as np
print('Imported Libraries')
```

```
Imported Libraries
```

Load Dataset

In [2]:

```python
df = pd.read_csv("E:\ML\dataset\project\dataset\price.csv")
df.head()
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19

5 rows × 26 columns

In [3]:

```python
df.tail()
```

Out[3]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	st
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	

Data Preparation

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```


In [6]:
```
df.isnull().sum()
# there are no null values
```

Out[6]:
```
car_ID            0
symboling         0
CarName           0
fueltype          0
aspiration        0
doornumber        0
carbody           0
drivewheel        0
enginelocation    0
wheelbase         0
carlength         0
carwidth          0
carheight         0
curbweight        0
enginetype        0
cylindernumber    0
enginesize        0
fuelsystem        0
boreratio         0
stroke            0
```

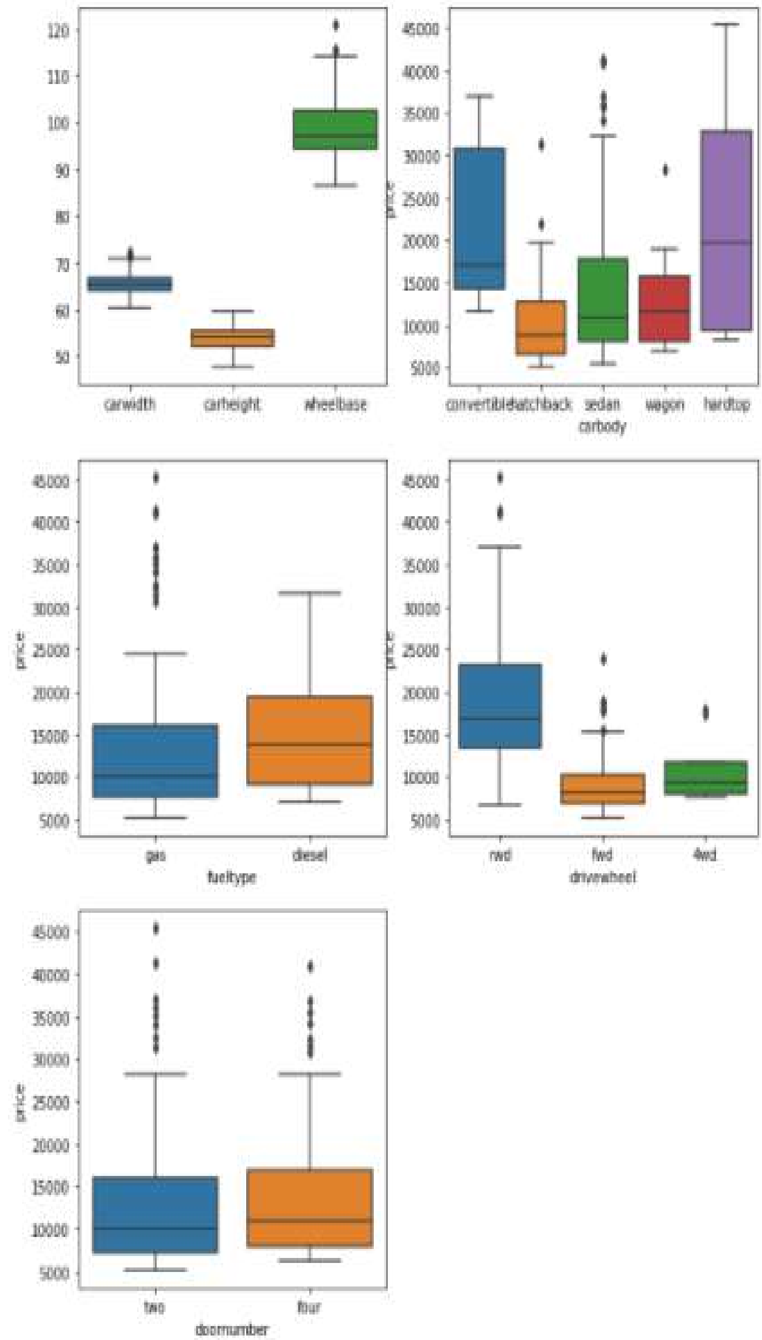
In [7]:
```
df['fueltype'].value_counts()
```

Out[7]:
```
gas       185
diesel     20
Name: fueltype, dtype: int64
```

In [8]:
```
df['carbody'].value_counts()
```
Out[8]:
```
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64
```
In [9]:
```
df['wheelbase'].value_counts().head()
```
Out[9]:
```
94.5    21
93.7    20
95.7    13
96.5     8
98.4     7
Name: wheelbase, dtype: int64
```
In [10]:
```
df['carlength'].value_counts().head()
```
Out[10]:
```
157.3    15
188.8    11
166.3     7
171.7     7
186.7     7
Name: carlength, dtype: int64
```
In [11]:
```
df['enginesize'].value_counts().head()
```
Out[11]:
```
122    15
92     15
98     14
97     14
108    13
Name: enginesize, dtype: int64
```
In [12]:
```
df['doornumber'].value_counts().head()
```
Out[12]:
```
four    115
two      90
Name: doornumber, dtype: int64
```

In [13]:

```
#Boxplot-display the summary of the set of data values having properties like minimum,
#first quartile, median, third quartile and maximum.

plt.figure(figsize=(10, 20))
plt.subplot(4,2,1)
df3=df[['carwidth','carheight','wheelbase']]
sns.boxplot(data=df3)
plt.subplot(4,2,2)
sns.boxplot(x = 'carbody', y = 'price', data = df)
plt.subplot(4,2,3)
sns.boxplot(x = 'fueltype', y = 'price', data = df)
plt.subplot(4,2,4)
sns.boxplot(x = 'drivewheel', y = 'price', data = df)
plt.subplot(4,2,5)
sns.boxplot(x = 'doornumber', y = 'price', data = df)
plt.show()
```



In [14]:
```python
df["CarName"].unique()
```

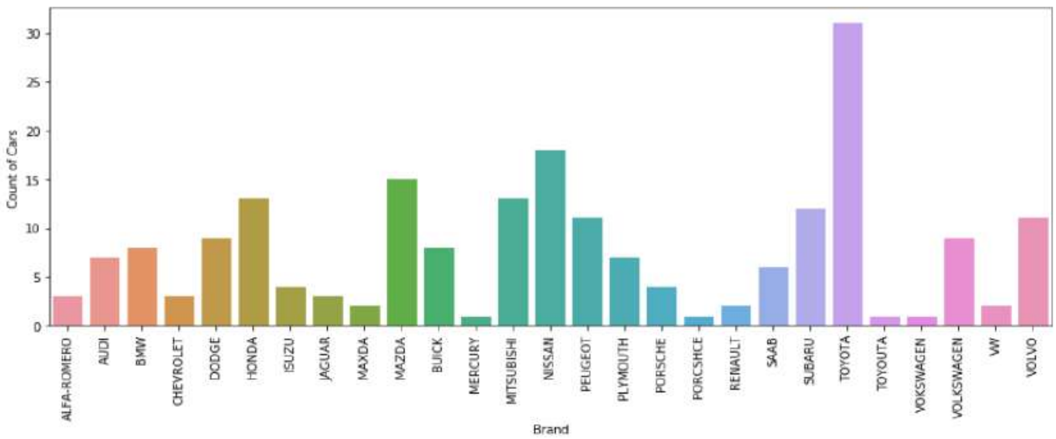
Out[14]:
```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
       'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
       'buick electra 225 custom', 'buick century luxus (sw)',
       'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
       'buick skylark', 'buick century special',
       'buick regal sport coupe (turbo)', 'mercury cougar',
       'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
       'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
       'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
       'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
       'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
       'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
       'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
       'peugeot 505s turbo diesel', 'plymouth fury iii',
       'plymouth cricket', 'plymouth satellite custom (sw)',
       'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
       'porsche macan', 'porcshce panamera', 'porsche cayenne',
       'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
       'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
       'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

In [15]:
```python
df['brand'] = df.CarName.str.split(' ').str.get(0).str.upper()
df['brand']
```

Out[15]:
```
0      ALFA-ROMERO
1      ALFA-ROMERO
2      ALFA-ROMERO
3             AUDI
4             AUDI
          ...
200          VOLVO
201          VOLVO
202          VOLVO
203          VOLVO
204          VOLVO
Name: brand, Length: 205, dtype: object
```

In [16]:
```python
fig, ax = plt.subplots(figsize = (15,5))
plt1 = sns.countplot(df['brand'], data=df)
plt1.set(xlabel = 'Brand', ylabel= 'Count of Cars')
xticks(rotation = 90)
```

Out[16]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26]),
 <a list of 27 Text xticklabel objects>)
```



Converting Categorical to numerical

In [17]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=le.fit_transform(df[i])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 27 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    int32
 3   fueltype          205 non-null    int32
 4   aspiration        205 non-null    int32
 5   doornumber        205 non-null    int32
 6   carbody           205 non-null    int32
 7   drivewheel        205 non-null    int32
 8   enginelocation    205 non-null    int32
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    int32
 15  cylindernumber    205 non-null    int32
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    int32
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
 26  brand             205 non-null    int32
dtypes: float64(8), int32(11), int64(8)
memory usage: 34.6 KB
```

In [18]:

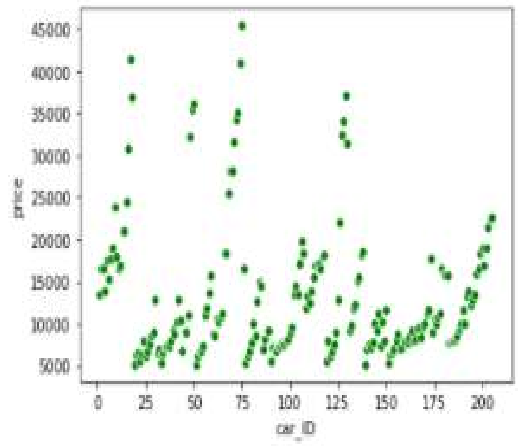
```
df.describe()
```

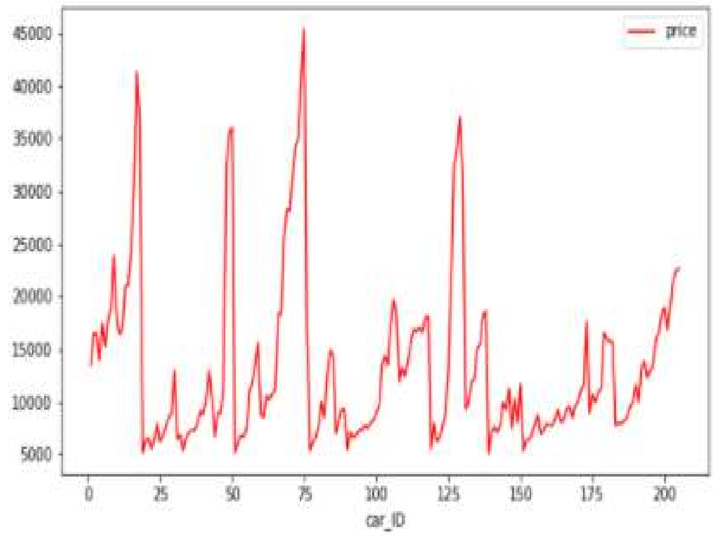
Out[18]:

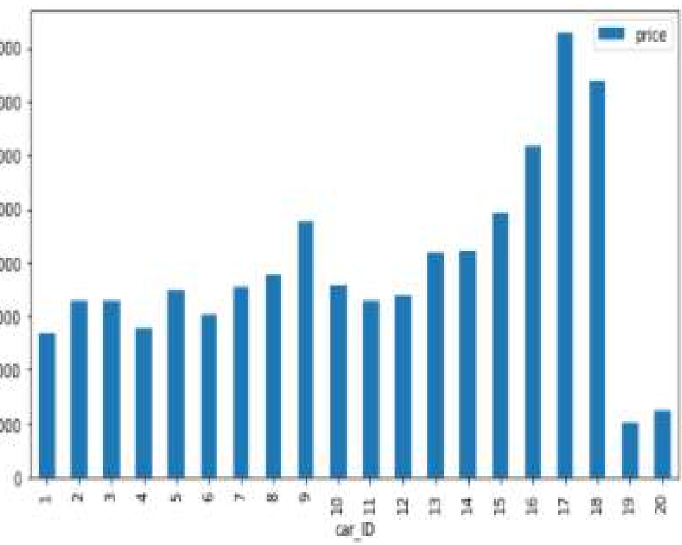
	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	fuelsystem	borer
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	...	205.000000	205.00C
mean	103.000000	0.834146	77.209756	0.902439	0.180488	0.439024	2.614634	1.326829	0.014634	98.756585	...	3.253659	3.329
std	59.322565	1.245307	41.014583	0.297446	0.385535	0.497483	0.859081	0.556171	0.120377	6.021776	...	2.013204	0.27C
min	1.000000	-2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	86.600000	...	0.000000	2.54C
25%	52.000000	0.000000	44.000000	1.000000	0.000000	0.000000	2.000000	1.000000	0.000000	94.500000	...	1.000000	3.15C

In [20]:

```
plt1 = sns.scatterplot(x = 'car_ID', y = 'price', data = df,color='green');

df.plot(x='car_ID',y='price',figsize=(9,5), color='red',);

df1.plot(x='car_ID',y='price',kind='bar',figsize=(9,5));
```

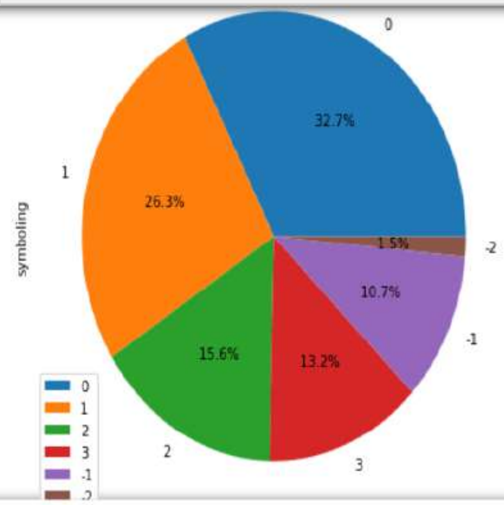






In [21]:

```python
#Its assigned insurance risk rating +3 high risk,-3 low risk
sym = pd.DataFrame(df['symboling'].value_counts())
print(sym)
sym.plot.pie(subplots=True,labels = sym.index.values, autopct='%5.1f%%', figsize = (7,7));

#There are more high risk vehicles
```

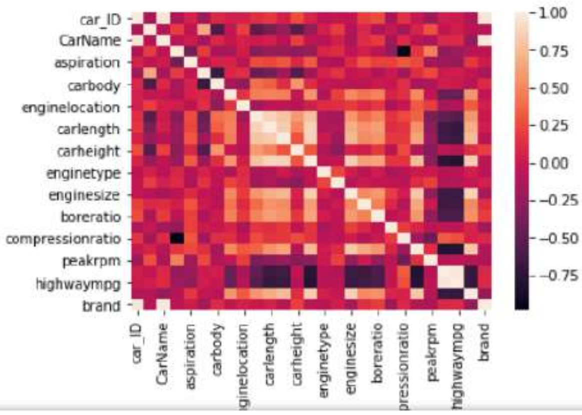


In [22]:

```python
#orrelation is a measure of how strongly one variable depends on another.
corr=df.corr()
sns.heatmap(corr)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x271f7cdb048>
```

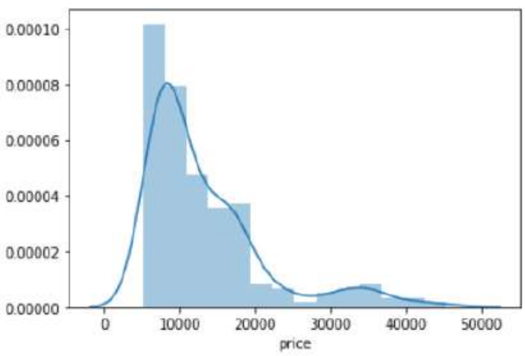


In [23]:

```python
#Target value(Histogram)
sns.distplot(df.price)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x271f7f45d08>
```



In [24]:
```
x=df.drop('price',axis=1) #independent variable
y=df['price'] #Dependent variable
```

Split the dataset

In [25]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 100)
```

Build the Regression model

1. Multiple Linear Regression

In [26]:
```
from sklearn.linear_model import LinearRegression
R1=LinearRegression()
R1.fit(x_train,y_train)
```

Out[26]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Predicted Values

In [27]:
```
y_pred1=R1.predict(x_test)
print("PredictValues:",y_pred1)
```

```
PredictValues: [ 6337.718017    9245.67272797 11095.51633448  8949.43565381
  7466.03237582 12178.78845564 15809.83707067 20095.84953432
 15858.21898828 21645.75791005 17643.24250926 15758.22505031
 19060.52657361 11819.01270301 39193.12004934  6865.01558932
  5489.16529513 13449.22157706 16536.65110517 15393.80880606
 16231.56443862 16182.87347234 34425.27979817  5467.48038844
 13453.32466474 21027.37070854 15604.08679807 27316.85752139
 14976.0232531  13686.82193287  6615.61225698 26872.39812969
 19212.72921957 19150.14515566 17322.11164274 10698.99050979
 15431.51760745 13012.60402687  6576.38155311 10120.47618787
 38515.85673178 14630.48502474  6209.62320389  9244.6053136
  6938.72937037 10890.25993461  7306.83813538 10193.23042643
  8136.86922466  8403.0036677   6423.0199223  13611.18920485
  6305.29331613 10173.26722007 20798.2066489   6360.06689424
 11499.745442   12896.74444303 15908.94849791  8136.73285003
  8790.65788407 34713.8129107 ]
```

Intercept and Coeff Values

In [28]:
```
print('The intercept value is ',R1.intercept_)
print()
print('The coeff values are',R1.coef_)

```

```
The intercept value is  -78199.22006249016

The coeff values are [ 3.93505149e+01  1.96630219e+02 -1.88376255e+01  4.07742587e+03
  4.24290820e+02 -3.85728894e+02 -4.80781887e+02  6.46133799e+02
  1.16824524e+04  8.88221522e+01 -4.91290508e+01  7.21695634e+02
  3.13292967e+02  3.33733827e+00  7.47837112e+01  5.72864877e+02
  6.77794263e+01  1.56911294e+02  3.34786270e+02 -1.94600051e+03
  3.51319196e+02  2.50917623e+01  9.03456495e-01 -9.75778257e+01
  1.33419974e+02 -3.98955062e+02]
```

The Actual Values vs Predicted Values

In [29]:
```
pd.DataFrame({'Actual Values':y_test,'Predicted Values':y_pred1})
```

Out[29]:

	Actual Values	Predicted Values
160	7738.0	6337.718017
186	8495.0	9245.672728
59	8845.0	11095.516334
165	9298.0	8949.435654
140	7603.0	7466.032376
...	...	...
28	8921.0	12896.744443
29	12964.0	15908.948498
182	7775.0	8136.732850
40	10295.0	8790.657884
128	37028.0	34713.812911

62 rows × 2 columns

Root mean square error

In [30]:
```
ms1=np.sqrt(mean_squared_error(y_test,y_pred1))
print('RMSE is ',ms1)
```

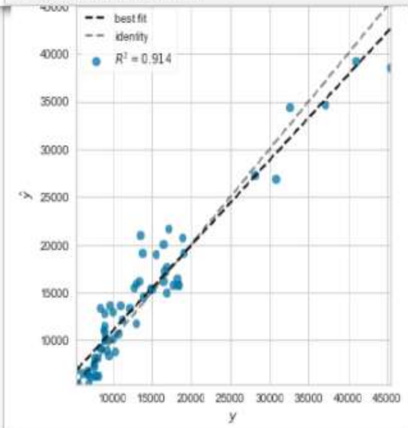
```
RMSE is  2447.942591346966
```

Score for the model

In [31]:
```
r1=R1.score(x_test,y_test)
print("r^2 score for Multiple regression for testing:",r1)
```

```
r^2 score for Multiple regression for testing: 0.9144295266645337
```

In [32]:
```
from yellowbrick.regressor import PredictionError

'''A prediction error plot shows the actual targets from the dataset
   against the predicted values generated by our model.
'''
visualizer = PredictionError(R1)
visualizer.fit(x_train, y_train)  # Fit the training data to the visualizer
visualizer.score(x_test, y_test)  # Evaluate the model on the test data
visualizer.poof() ;
```



2. Decision Tree Regression

In [33]:
```
from sklearn.tree import DecisionTreeRegressor
R2=DecisionTreeRegressor()
R2.fit(x_train,y_train)
```

Out[33]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

Predicted Values

In [34]:
```
y_pred2=R2.predict(x_test)
y_pred2
```

Out[34]:
```
array([ 9258.,  8195., 10595.,  9538.,  7053.,  8495., 18950., 21105.,
        11549., 19699., 22470., 18620., 16695.,  8845., 34184.,  7129.,
         9258., 11850., 21105., 12629., 15985., 15985., 34028.,  5389.,
        11199., 14399., 16500., 25552., 16430., 11900.,  6669., 41315.,
        17950., 22470., 11900., 10795., 11850., 11199.,  7299., 10245.,
        34184., 16430.,  8358.,  9980.,  6479.,  9549.,  6692.,  9279.,
         7788.,  7788.,  6095., 11199.,  8058.,  8189., 17710.,  7788.,
        18344., 18344., 12629.,  7788.,  8845., 34028.])
```

The Actual Values vs Predicted Values

In [35]:
```
pd.DataFrame({'Actual Values':y_test,'Predicted Values':y_pred2})
```

Out[35]:

	Actual Values	Predicted Values
160	7738.0	9258.0
186	8495.0	8195.0
59	8845.0	10595.0
165	9298.0	9538.0
140	7603.0	7053.0
...	...	...
28	8921.0	18344.0
29	12964.0	12629.0
182	7775.0	7788.0
40	10295.0	8845.0
128	37028.0	34028.0

62 rows × 2 columns

Root mean square error

In [36]:
```
ms2=np.sqrt(mean_squared_error(y_test,y_pred2))
print('RMSE is ',ms2)
```

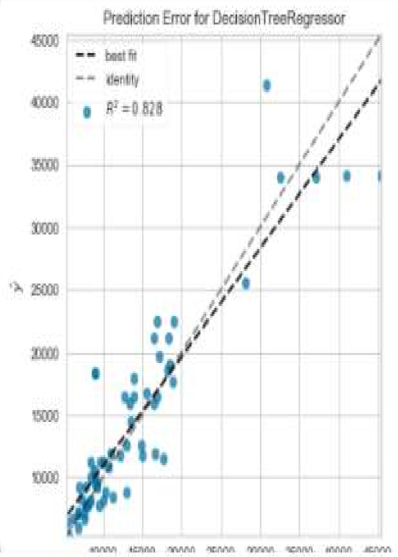
```
RMSE is  3467.8313875184394
```

Score for the model

In [56]:
```
r2=R2.score(x_test,y_test)
print("r^2 score for Decision Tree regressor for testing:",r2)
```

```
r^2 score for Decision Tree regressor for testing: 0.8282734585795191
```

In [38]:
```
visualizer = PredictionError(R2)
visualizer.fit(x_train, y_train)  # Fit the training data to the visualizer
visualizer.score(x_test, y_test)  # Evaluate the model on the test data
visualizer.poof() ;
```



3. Random Forest Regressor

In [39]:
```
from sklearn.ensemble import RandomForestRegressor
R3=RandomForestRegressor()
R3.fit(x_train,y_train)
```

Out[39]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

Predicted Values

In [40]:
```
y_pred3=R3.predict(x_test)
y_pred3
```

Out[40]:
```
array([ 7845.69   ,  8948.73   ,  9662.96   ,  9596.31   ,  7498.49   ,
       10192.65   , 18088.18334, 18174.63336, 13287.92167, 18374.96   ,
       19504.79167, 17757.53167, 14165.1    , 11223.47   , 34332.135  ,
        6819.045  ,  7974.535  , 14314.52   , 16404.07   , 14452.53167,
       15299.05   , 16085.67   , 31250.965  ,  5860.775  , 11756.98   ,
       14815.39   , 14697.67167, 27175.18834, 13781.51   , 12999.56167,
        6235.26   , 35871.48167, 17382.25   , 20727.37501, 13700.01   ,
       10947.2    , 14635.84   , 11835.35   ,  7068.65   , 10203.11   ,
       35193.855  , 11617.25   ,  7881.83   ,  8227.13   ,  6097.26   ,
        9416.93   ,  6695.3    ,  9564.83   ,  8253.855  ,  8693.335  ,
        5973.255  , 11769.64   ,  8040.43   , 10270.79   , 19926.89167,
        8083.92   , 11392.19   , 12682.28   , 15088.42835,  8221.315  ,
        9275.965  , 31302.365  ])
```

The Actual Values vs Predicted Values

In [41]:
```
pd.DataFrame({'Actual Values':y_test,'Predicted Values':y_pred3})
```

Out[41]:

	Actual Values	Predicted Values
160	7738.0	7845.69000
186	8495.0	8948.73000
59	8845.0	9662.96000
165	9298.0	9596.31000
140	7603.0	7498.49000
...	...	...
28	8921.0	12682.28000
29	12964.0	15088.42835
182	7775.0	8221.31500
40	10295.0	9275.96500
128	37028.0	31302.36500

62 rows × 2 columns

Root mean square error

In [42]:
```
ms3=np.sqrt(mean_squared_error(y_test,y_pred3))
print('RMSE is ',ms3)
```

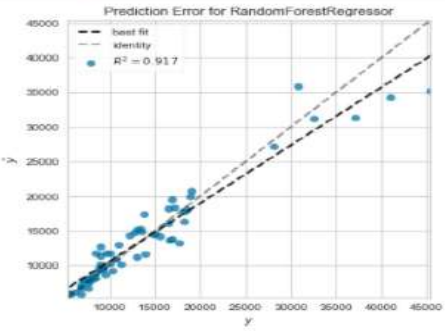
```
RMSE is  2411.2505468363215
```

Score for the model

In [57]:
```
r3=R3.score(x_test,y_test)
print("r^2 score for RandomForestRegressor for testing:",r3)
```

```
r^2 score for RandomForestRegressor for testing: 0.9169755216858728
```

In [44]:
```
visualizer = PredictionError(R3)
visualizer.fit(x_train, y_train)  # Fit the training data to the visualizer
visualizer.score(x_test, y_test)  # Evaluate the model on the test data
visualizer.poof() ;
```



4. Ridge Regressor

In [45]:
```
from sklearn.linear_model import Ridge
R4 = Ridge(alpha=0.01)
R4.fit(x_train, y_train)
```

Out[45]:
```
Ridge(alpha=0.01, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

Predicted Values

In [46]:
```
y_pred4=R4.predict(x_test)
y_pred4
```

Out[46]:
```
array([ 6351.5356953 ,  9249.32642024, 11097.30478539,  8962.45161113,
        7471.11026756, 12187.15822245, 15832.2065654 , 20096.54103683,
       15843.49585735, 21660.38847656, 17627.91643351, 15749.11725679,
       19066.67590985, 11810.57864984, 39215.89188713,  6873.22891401,
        5495.92706634, 13444.34589235, 16550.40646704, 15398.98570782,
       16218.62181992, 16169.67807411, 34310.46189332,  5466.2893685 ,
       13445.8697097 , 21038.35226979, 15619.72741513, 27316.29623808,
       14985.17160132, 13650.22668565,  6608.8041099 , 26919.06513728,
       19210.8842874 , 19140.09042641, 17310.44164444, 10685.48596988,
       15451.80761915, 13001.61387122,  6575.34526278, 10111.0266675 ,
       38548.01555294, 14594.09497892,  6219.40387736,  9252.00878273,
        6912.43886521, 10915.60735919,  7295.43948669, 10227.28044398,
        8150.36196329,  8388.07033774,  6424.12037956, 13604.52146264,
        6312.55014269, 10156.02750015, 20785.10272361,  6371.27479894,
       11507.96831426, 12896.19912725, 15917.50763806,  8145.77585176,
        8799.51048994, 34600.91937247])
```

The Actual Values vs Predicted Values

In [47]:
```
pd.DataFrame({'Actual Values':y_test,'Predicted Values':y_pred4})
```

Out[47]:

	Actual Values	Predicted Values
160	7738.0	6351.535695
186	8495.0	9249.326420
59	8845.0	11097.304785
165	9298.0	8962.451611
140	7603.0	7471.110268
...	...	...
28	8921.0	12896.199127
29	12964.0	15917.507638
182	7775.0	8145.775852
40	10295.0	8799.510490
128	37028.0	34600.919372

62 rows × 2 columns

Root mean square error

In [48]:
```
ms4=np.sqrt(mean_squared_error(y_test,y_pred4))
print('RMSE is ',ms4)
```

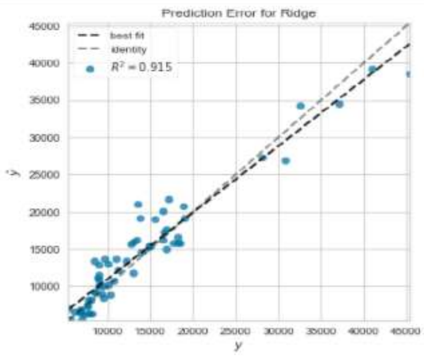
```
RMSE is  2444.75556590593
```

Score for the model

In [58]:
```
r4=R4.score(x_test,y_test)
print("r^2 score for Ridge Regressor for testing:",r4)
```

```
r^2 score for Ridge Regressor for testing: 0.914652193445812
```

In [50]:
```
visualizer = PredictionError(R4)
visualizer.fit(x_train, y_train)  # Fit the training data to the visualizer
visualizer.score(x_test, y_test)  # Evaluate the model on the test data
visualizer.poof() ;
```



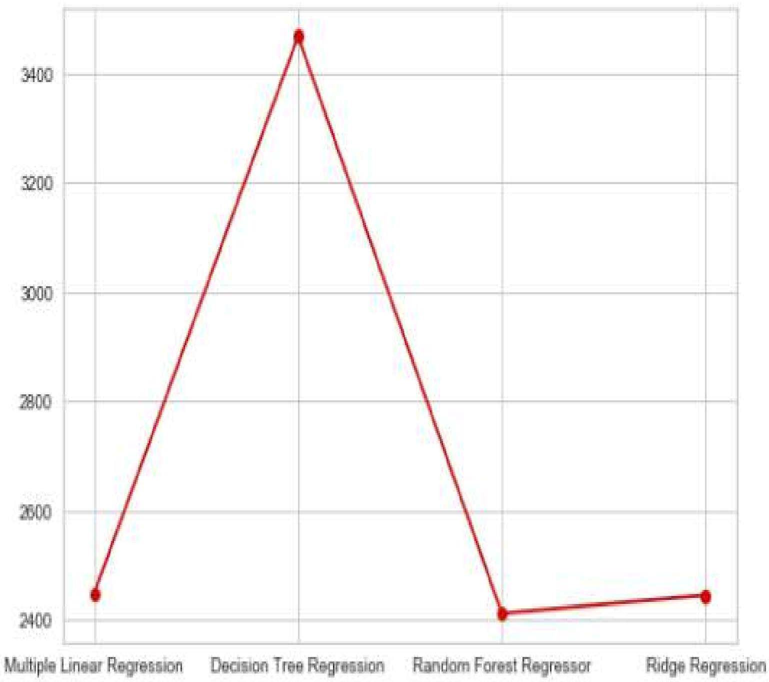
In [51]:
```
model=['Multiple Linear Regression','Decision Tree Regression','Random Forest Regressor','Ridge Regression']
rmse=[ms1,ms2,ms3,ms4]
r2score=[r1,r2,r3,r4]
table=pd.DataFrame(data=zip(model,rmse,r2score),columns=['Model','RSME Value','R2 Score'])
table
```

Out[51]:

	Model	RSME Value	R2 Score
0	Multiple Linear Regression	2447.942591	0.914430
1	Decision Tree Regression	3407.831388	0.828273
2	Random Forest Regressor	2411.250547	0.916976
3	Ridge Regression	2444.755566	0.914652

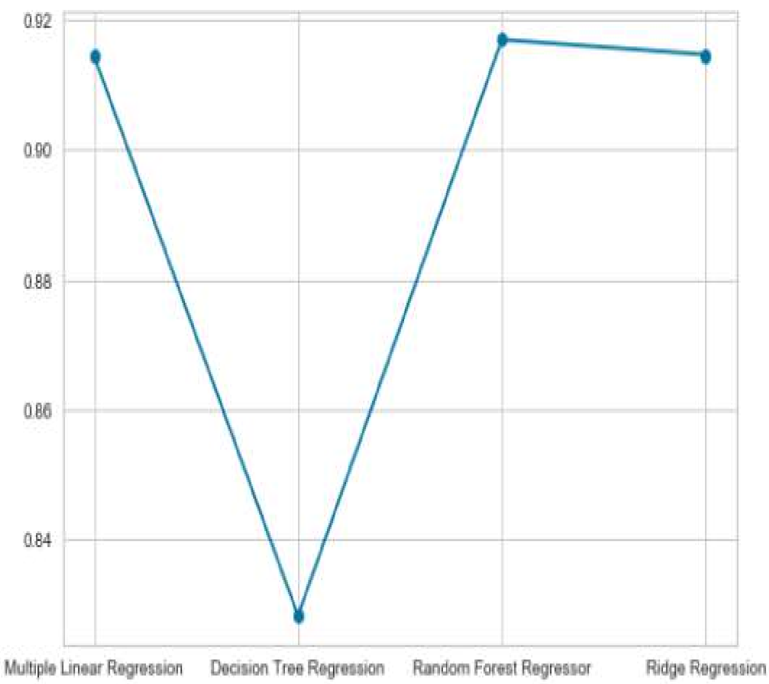
```
In [52]:  1 plt.plot(model,rmse,'ro-')
```

```
Out[52]: [<matplotlib.lines.Line2D at 0x271f8b72888>]
```



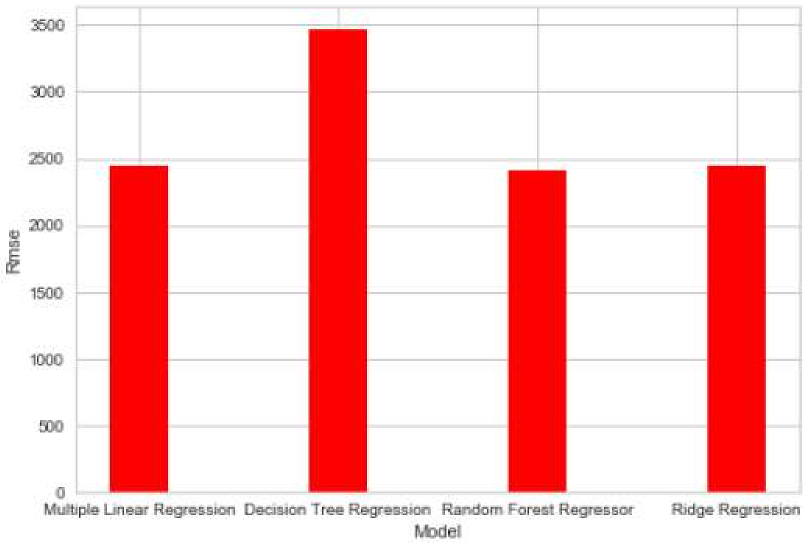

```
In [53]:  1 plt.plot(model,r2score,'bo-')
```

```
Out[53]: [<matplotlib.lines.Line2D at 0x271f81e6648>]
```

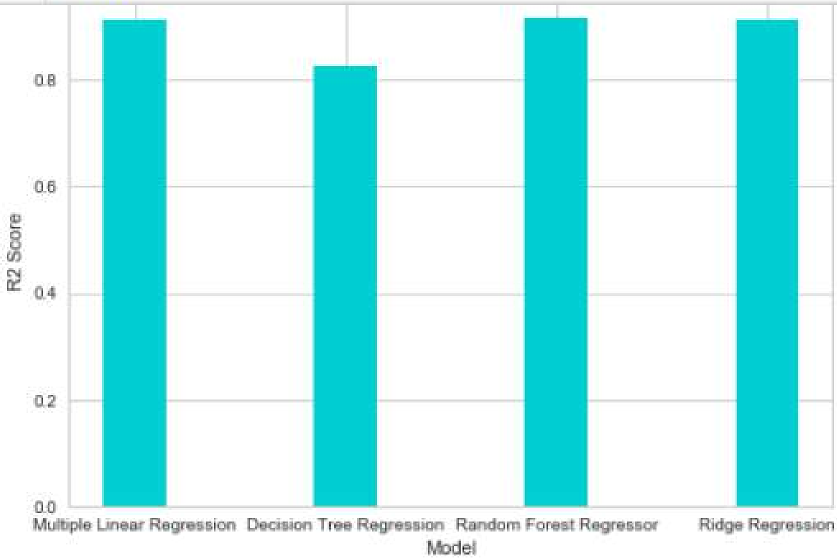


In [54]:
```
plt.bar(model,rmse,width=0.298,color='red')
plt.xlabel('Model')
plt.ylabel('Rmse');

print('Decision Tree Regression has Maximum Rmse value',ms2)
print()
```

```
Decision Tree Regression has Maximum Rmse value 3467.8313875184394
```



In [55]:
```
plt.bar(model,r2score,width=0.298,color='darkturquoise')
plt.xlabel('Model')
plt.ylabel('R2 Score');

print('Random Forest Regression has Maximum R2 Score value',r3)
print()
```



CONCLUSION

We have predicted the prices of the car. The different types of regression analysis techniques get used when the target and independent variables show a linear or non-linear relationship between each other, and the target variable contains continuous values. The results obtained thus conclude that Decision Tree Regression has Maximum RMSE value ie 3467.83 and Random Forest Regression has Maximum R2 Score value 0.916.

The Random Forest regression is an ensemble learning method which combines multiple decision trees and predicts the final output based on the average of each tree output. With the help of Random Forest regression, we can prevent Over fitting in the model by creating random subsets of the dataset.

A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used. So the Ridge Regression has better R2 Score compared with Multiple Linear Regression.

	Model	RSME Value	R2 Score
0	Multiple Linear Regression	2447.942591	0.914430
1	Decision Tree Regression	3467.831388	0.828273
2	Random Forest Regressor	2411.250547	0.916976
3	Ridge Regression	2444.755566	0.914652

REFERENCES

1) https://www.kaggle.com/goyalshalini93/car-price-prediction-linear-regression-rfe
2) N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used car by using regression models," 2018 5th International Conference on Business and Industrial Research (ICBIR), Bangkok, 2018, pp. 115-119.
3) Listiani M. 2009. Support Vector Regression Analysis for Price Prediction in a Car Leasing Application. Master Thesis. Hamburg University of Technology