ylwmvn0po

January 13, 2023

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[6]:
```
hello =pd.read_csv(r'/content/car.txt')
hello.head()
```

[6]:

	name	company	year	Price	\
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	

	kms_driven	fuel_type
0	45,000 kms	Petrol
1	40 kms	Diesel
2	22,000 kms	Petrol
3	28,000 kms	Petrol
4	36,000 kms	Diesel

[7]:
```
hello.shape
```

[7]:
```
(892, 6)
```

[8]:
```
hello.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
```

```
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: object(6)
memory usage: 41.9+ KB
```

[9]:
```
hello.describe()
```

[9]:

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

[10]:
```
hello['year'].unique()
```

[10]:
```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '…', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o…', 'ture', 'emi', 'car', 'able', 'no.',
       'd…', 'SALE', 'digo', 'sell', 'd Ex', 'n…', 'e…', 'D…',
       ', Ac', 'go .', 'k…', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r…', 'zest'], dtype=object)
```

[11]:
```
hello['year'].value_counts()
```

[11]:
```
2015    117
2014     94
2013     94
2016     76
2012     75
       …
ture      1
emi       1
able      1
no.       1
zest      1
Name: year, Length: 61, dtype: int64
```

[12]:
```
hello['Price'].unique()
```

[12]:
```
array(['80,000', '4,25,000', 'Ask For Price', '3,25,000', '5,75,000',
       '1,75,000', '1,90,000', '8,30,000', '2,50,000', '1,82,000',
       '3,15,000', '4,15,000', '3,20,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,90,000', '95,000', '1,80,000', '3,85,000', '1,05,000',
       '6,50,000', '6,89,999', '4,48,000', '5,49,000', '5,01,000',
```

'4,89,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
'4,99,999', '2,35,000', '2,49,999', '14,75,000', '3,95,000',
'2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
'1,10,000', '4,48,999', '18,91,111', '1,59,500', '3,44,999',
'4,49,999', '8,65,000', '6,99,000', '3,75,000', '2,24,999',
'12,00,000', '1,95,000', '3,51,000', '2,40,000', '90,000',
'1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,90,000',
'1,35,000', '16,00,000', '7,01,000', '2,65,000', '5,25,000',
'3,72,000', '6,35,000', '5,50,000', '4,85,000', '3,29,500',
'2,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
'4,50,000', '2,70,000', '1,58,400', '1,79,000', '1,25,000',
'2,99,000', '1,50,000', '2,75,000', '2,85,000', '3,40,000',
'70,000', '2,89,999', '8,49,999', '7,49,999', '2,74,999',
'9,84,999', '5,99,999', '2,44,999', '4,74,999', '2,45,000',
'1,69,500', '3,70,000', '1,68,000', '1,45,000', '98,500',
'2,09,000', '1,85,000', '9,00,000', '6,99,999', '1,99,999',
'5,44,999', '1,99,000', '5,40,000', '49,000', '7,00,000', '55,000',
'8,95,000', '3,55,000', '5,65,000', '3,65,000', '40,000',
'4,00,000', '3,30,000', '5,80,000', '3,79,000', '2,19,000',
'5,19,000', '7,30,000', '20,00,000', '21,00,000', '14,00,000',
'3,11,000', '8,55,000', '5,35,000', '1,78,000', '3,00,000',
'2,55,000', '5,49,999', '3,80,000', '57,000', '4,10,000',
'2,25,000', '1,20,000', '59,000', '5,99,000', '6,75,000', '72,500',
'6,10,000', '2,30,000', '5,20,000', '5,24,999', '4,24,999',
'6,44,999', '5,84,999', '7,99,999', '4,44,999', '6,49,999',
'9,44,999', '5,74,999', '3,74,999', '1,30,000', '4,01,000',
'13,50,000', '1,74,999', '2,39,999', '99,999', '3,24,999',
'10,74,999', '11,30,000', '1,49,000', '7,70,000', '30,000',
'3,35,000', '3,99,000', '65,000', '1,69,999', '1,65,000',
'5,60,000', '9,50,000', '7,15,000', '45,000', '9,40,000',
'1,55,555', '15,00,000', '4,95,000', '8,00,000', '12,99,000',
'5,30,000', '14,99,000', '32,000', '4,05,000', '7,60,000',
'7,50,000', '4,19,000', '1,40,000', '15,40,000', '1,23,000',
'4,98,000', '4,80,000', '4,88,000', '15,25,000', '5,48,900',
'7,25,000', '99,000', '52,000', '28,00,000', '4,99,000',
'3,81,000', '2,78,000', '6,90,000', '2,60,000', '90,001',
'1,15,000', '15,99,000', '1,59,000', '51,999', '2,15,000',
'35,000', '11,50,000', '2,69,000', '60,000', '4,30,000',
'85,00,003', '4,01,919', '4,90,000', '4,24,000', '2,05,000',
'5,49,900', '3,71,500', '4,35,000', '1,89,700', '3,89,700',
'3,60,000', '2,95,000', '1,14,990', '10,65,000', '4,70,000',
'48,000', '1,88,000', '4,65,000', '1,79,999', '21,90,000',
'23,90,000', '10,75,000', '4,75,000', '10,25,000', '6,15,000',
'19,00,000', '14,90,000', '15,10,000', '18,50,000', '7,90,000',
'17,25,000', '12,25,000', '68,000', '9,70,000', '31,00,000',
'8,99,000', '88,000', '53,000', '5,68,500', '71,000', '5,90,000',
'7,95,000', '42,000', '1,89,000', '1,62,000', '35,999',

```
       '29,00,000', '39,999', '50,500', '5,10,000', '8,60,000',
       '5,00,001'], dtype=object)
```

[13]:
```
hello['Price'].nunique()
```

[13]:
```
274
```

[14]:
```
hello['Price'].value_counts()
```

[14]:
```
Ask For Price    35
2,50,000         17
3,50,000         14
1,80,000         13
1,30,000         12
                 ..
7,49,999          1
11,30,000         1
10,74,999         1
3,24,999          1
5,00,001          1
Name: Price, Length: 274, dtype: int64
```

[15]:
```
hello.isnull().sum()
```

[15]:
```
name           0
company        0
year           0
Price          0
kms_driven    52
fuel_type     55
dtype: int64
```

[16]:
```
hello['kms_driven'].isna()
```

[16]:
```
0      False
1      False
2      False
3      False
4      False
       …
887     True
888    False
889    False
890    False
891    False
Name: kms_driven, Length: 892, dtype: bool
```

[17]:
```
hello['year'].unique()
```

[17]:
```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '…', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o…', 'ture', 'emi', 'car', 'able', 'no.',
       'd…', 'SALE', 'digo', 'sell', 'd Ex', 'n…', 'e…', 'D…',
       ', Ac', 'go .', 'k…', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r…', 'zest'], dtype=object)
```

[18]:
```
hello=hello[hello['year'].str.isnumeric()]
hello.head()
```

[18]:

	name	company	year	Price	\
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	

	kms_driven	fuel_type
0	45,000 kms	Petrol
1	40 kms	Diesel
2	22,000 kms	Petrol
3	28,000 kms	Petrol
4	36,000 kms	Diesel

[19]:
```
hello['year']=hello['year'].astype(int)
hello['year']
```

```
<ipython-input-19-b93993034a86>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  hello['year']=hello['year'].astype(int)
```

[19]:
```
0      2007
1      2006
2      2018
3      2014
4      2014
       …
886    2009
888    2018
```

```
889    2013
890    2014
891    2014
Name: year, Length: 842, dtype: int64
```

[20]:
```
hello.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 842 entries, 0 to 891
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         842 non-null    object
 1   company      842 non-null    object
 2   year         842 non-null    int64
 3   Price        842 non-null    object
 4   kms_driven   840 non-null    object
 5   fuel_type    837 non-null    object
dtypes: int64(1), object(5)
memory usage: 46.0+ KB
```

[21]:
```
hello['year'].unique()
```

[21]:
```
array([2007, 2006, 2018, 2014, 2015, 2012, 2013, 2016, 2010, 2017, 2008,
       2011, 2019, 2009, 2005, 2000, 2003, 2004, 1995, 2002, 2001])
```

[22]:
```
hello['Price'].unique()
```

[22]:
```
array(['80,000', '4,25,000', 'Ask For Price', '3,25,000', '5,75,000',
       '1,75,000', '1,90,000', '8,30,000', '2,50,000', '1,82,000',
       '3,15,000', '4,15,000', '3,20,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,90,000', '95,000', '1,80,000', '3,85,000', '1,05,000',
       '6,50,000', '6,89,999', '4,48,000', '5,49,000', '5,01,000',
       '4,89,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
       '4,99,999', '2,35,000', '2,49,999', '14,75,000', '3,95,000',
       '2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
       '1,10,000', '4,48,999', '18,91,111', '1,59,500', '3,44,999',
       '4,49,999', '8,65,000', '6,99,000', '3,75,000', '2,24,999',
       '12,00,000', '1,95,000', '3,51,000', '2,40,000', '90,000',
       '1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,90,000',
       '1,35,000', '16,00,000', '7,01,000', '2,65,000', '5,25,000',
       '3,72,000', '6,35,000', '5,50,000', '4,85,000', '3,29,500',
       '2,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
       '4,50,000', '2,70,000', '1,58,400', '1,79,000', '1,25,000',
       '2,99,000', '1,50,000', '2,75,000', '2,85,000', '3,40,000',
       '70,000', '2,89,999', '8,49,999', '7,49,999', '2,74,999',
```

```
       '9,84,999', '5,99,999', '2,44,999', '4,74,999', '2,45,000',
       '1,69,500', '3,70,000', '1,68,000', '1,45,000', '98,500',
       '2,09,000', '1,85,000', '9,00,000', '6,99,999', '1,99,999',
       '5,44,999', '1,99,000', '5,40,000', '49,000', '7,00,000', '55,000',
       '8,95,000', '3,55,000', '5,65,000', '3,65,000', '40,000',
       '4,00,000', '3,30,000', '5,80,000', '3,79,000', '2,19,000',
       '5,19,000', '7,30,000', '20,00,000', '21,00,000', '14,00,000',
       '3,11,000', '8,55,000', '5,35,000', '1,78,000', '3,00,000',
       '2,55,000', '5,49,999', '3,80,000', '57,000', '4,10,000',
       '2,25,000', '1,20,000', '59,000', '5,99,000', '6,75,000', '72,500',
       '6,10,000', '2,30,000', '5,20,000', '5,24,999', '4,24,999',
       '6,44,999', '5,84,999', '7,99,999', '4,44,999', '6,49,999',
       '9,44,999', '5,74,999', '3,74,999', '1,30,000', '4,01,000',
       '13,50,000', '1,74,999', '2,39,999', '99,999', '3,24,999',
       '10,74,999', '11,30,000', '1,49,000', '7,70,000', '30,000',
       '3,35,000', '3,99,000', '65,000', '1,69,999', '1,65,000',
       '5,60,000', '9,50,000', '7,15,000', '45,000', '9,40,000',
       '1,55,555', '15,00,000', '4,95,000', '8,00,000', '12,99,000',
       '5,30,000', '14,99,000', '32,000', '4,05,000', '7,60,000',
       '7,50,000', '4,19,000', '1,40,000', '15,40,000', '1,23,000',
       '4,98,000', '4,80,000', '4,88,000', '15,25,000', '5,48,900',
       '7,25,000', '99,000', '52,000', '28,00,000', '4,99,000',
       '3,81,000', '2,78,000', '6,90,000', '2,60,000', '90,001',
       '1,15,000', '15,99,000', '1,59,000', '51,999', '2,15,000',
       '35,000', '11,50,000', '2,69,000', '60,000', '4,30,000',
       '85,00,003', '4,01,919', '4,90,000', '4,24,000', '2,05,000',
       '5,49,900', '4,35,000', '1,89,700', '3,89,700', '3,60,000',
       '2,95,000', '1,14,990', '10,65,000', '4,70,000', '48,000',
       '1,88,000', '4,65,000', '1,79,999', '21,90,000', '23,90,000',
       '10,75,000', '4,75,000', '10,25,000', '6,15,000', '19,00,000',
       '14,90,000', '15,10,000', '18,50,000', '7,90,000', '17,25,000',
       '12,25,000', '68,000', '9,70,000', '31,00,000', '8,99,000',
       '88,000', '53,000', '5,68,500', '71,000', '5,90,000', '7,95,000',
       '42,000', '1,89,000', '1,62,000', '35,999', '29,00,000', '39,999',
       '50,500', '5,10,000', '8,60,000', '5,00,001'], dtype=object)
```

[23]:
```
hello=hello[hello['Price']!='Ask For Price']
hello.head()
```

[23]:

	name	company	year	Price	\
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	
6	Ford Figo	Ford	2012	1,75,000	

	kms_driven	fuel_type

```
0  45,000 kms    Petrol
1      40 kms    Diesel
3  28,000 kms    Petrol
4  36,000 kms    Diesel
6  41,000 kms    Diesel
```

[24]:
```
hello['Price']=hello['Price'].str.replace(',','').astype(int)
hello['Price']
```

[24]:
```
0       80000
1      425000
3      325000
4      575000
6      175000
        ...
886    300000
888    260000
889    390000
890    180000
891    160000
Name: Price, Length: 819, dtype: int64
```

[25]:
```
hello.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 819 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        819 non-null    object
 1   company     819 non-null    object
 2   year        819 non-null    int64 
 3   Price       819 non-null    int64 
 4   kms_driven  819 non-null    object
 5   fuel_type   816 non-null    object
dtypes: int64(2), object(4)
memory usage: 44.8+ KB
```

[26]:
```
hello['kms_driven'].isnull().sum()
```

[26]:
```
0
```

[27]:
```
hello.isnull().sum()
```

[27]:
```
name          0
company       0
year          0
```

```
Price         0
kms_driven    0
fuel_type     3
dtype: int64
```

[28]:
```
hello['kms_driven'].value_counts()
```

[28]:
```
45,000 kms      30
35,000 kms      29
55,000 kms      25
50,000 kms      23
20,000 kms      21
                ..
1,00,200 kms     1
65 kms           1
30,874 kms       1
1,03,553 kms     1
72,160 kms       1
Name: kms_driven, Length: 250, dtype: int64
```

[29]:
```
hello['kms_driven'].str.isnumeric()
```

[29]:
```
0      False
1      False
3      False
4      False
6      False
       …
886    False
888    False
889    False
890    False
891    False
Name: kms_driven, Length: 819, dtype: bool
```

[30]:
```
hello['kms_driven'].str.get(0)
```

[30]:
```
0      4
1      4
3      2
4      3
6      4
      ..
886    1
888    2
889    4
890    P
```

```
891    P
Name: kms_driven, Length: 819, dtype: object
```

[31]:
```
hello['kms_driven']=hello['kms_driven'].str.split(' ').str.get(0).str.
 ↪replace(',','')
hello['kms_driven']
```

[31]:
```
0       45000
1          40
3       28000
4       36000
6       41000
        …
886    132000
888     27000
889     40000
890    Petrol
891    Petrol
Name: kms_driven, Length: 819, dtype: object
```

[32]:
```
hello=hello[hello['kms_driven'].str.isnumeric()]
hello.head()
```

[32]:

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	425000	40	Diesel
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	575000	36000	Diesel
6	Ford Figo	Ford	2012	175000	41000	Diesel

[33]:
```
hello['kms_driven']=hello['kms_driven'].astype(int)
hello['kms_driven']
```

```
<ipython-input-33-da731c0deedc>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  hello['kms_driven']=hello['kms_driven'].astype(int)
```

[33]:
```
0       45000
1          40
3       28000
4       36000
6       41000
        ...
883     50000
885     30000
886    132000
888     27000
889     40000
Name: kms_driven, Length: 817, dtype: int64
```

[34]:
```
hello.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 817 entries, 0 to 889
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        817 non-null    object
 1   company     817 non-null    object
 2   year        817 non-null    int64
 3   Price       817 non-null    int64
 4   kms_driven  817 non-null    int64
 5   fuel_type   816 non-null    object
dtypes: int64(3), object(3)
memory usage: 44.7+ KB
```

[35]:
```
hello['kms_driven'].isnull().sum()
```

[35]:
```
0
```

[36]:
```
hello=hello[~hello['fuel_type'].isna()]
hello.head()
```

[36]:

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	425000	40	Diesel
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	575000	36000	
6	Ford Figo	Ford	2012	175000	41000	

```
4    Diesel
6    Diesel
```

[37]:
```
hello['company'].value_counts().sum()
```

[37]:
```
816
```

[38]:
```
hello['company'].unique()
```

[38]:
```
array(['Hyundai', 'Mahindra', 'Ford', 'Maruti', 'Skoda', 'Audi', 'Toyota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswagen',
       'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat', 'Force',
       'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'], dtype=object)
```

[39]:
```
hello['company']
```

[39]:
```
0       Hyundai
1      Mahindra
3       Hyundai
4          Ford
6          Ford
         ...
883      Maruti
885        Tata
886      Toyota
888        Tata
889    Mahindra
Name: company, Length: 816, dtype: object
```

[40]:
```
hello.describe()
```

[40]:

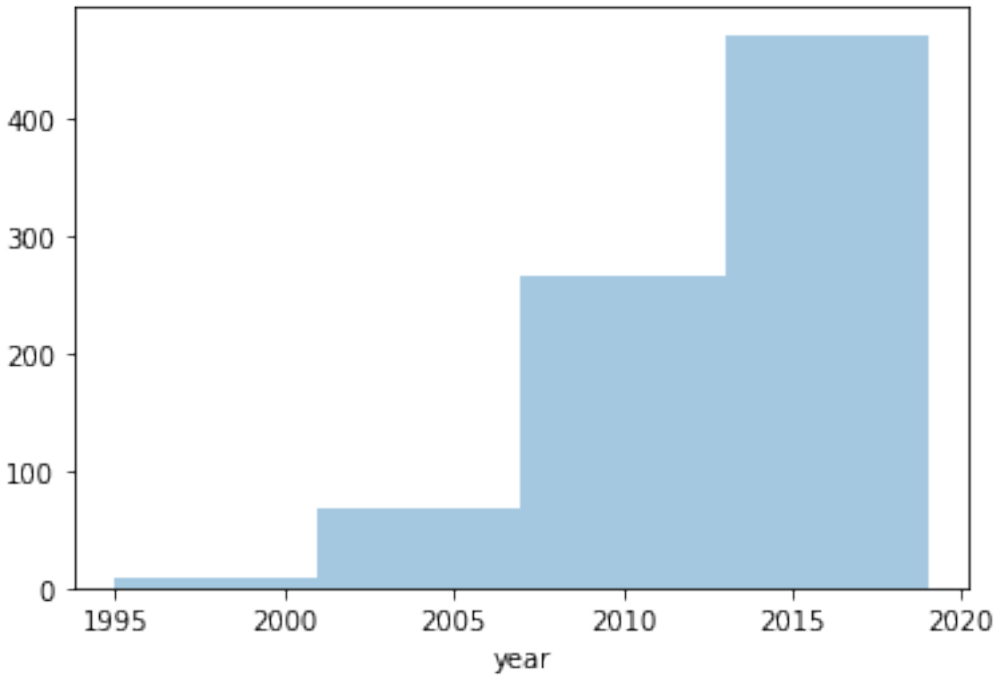
	year	Price	kms_driven
count	816.000000	8.160000e+02	816.000000
mean	2012.444853	4.117176e+05	46275.531863
std	4.002992	4.751844e+05	34297.428044
min	1995.000000	3.000000e+04	0.000000
25%	2010.000000	1.750000e+05	27000.000000
50%	2013.000000	2.999990e+05	41000.000000
75%	2015.000000	4.912500e+05	56818.500000
max	2019.000000	8.500003e+06	400000.000000

[41]:
```
sns.distplot(hello['year'],kde=False,bins=4)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
```

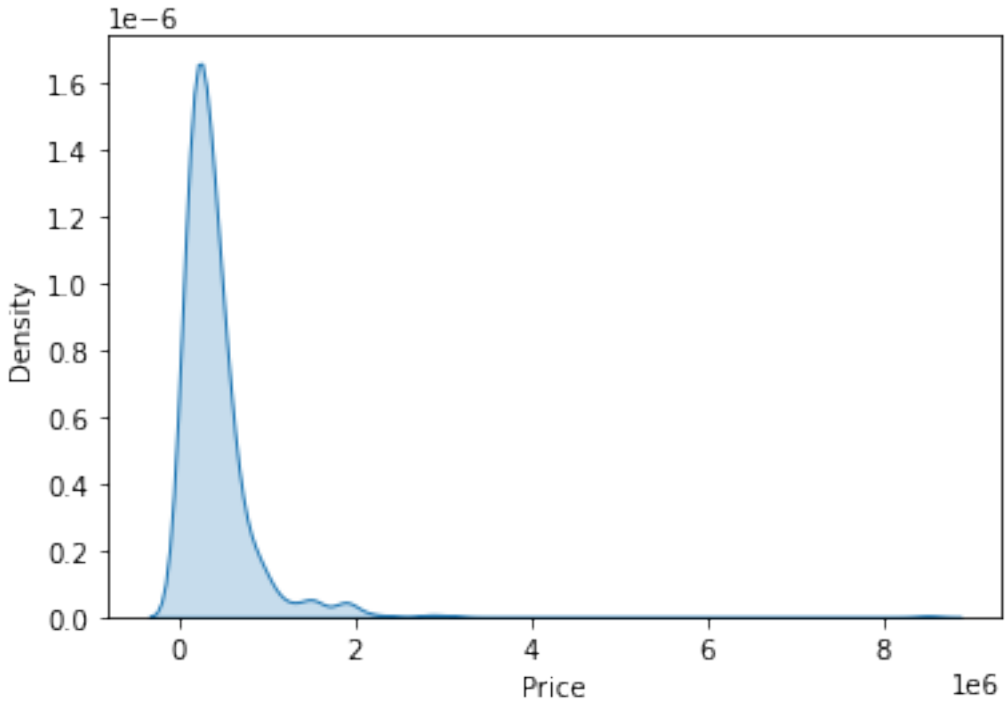
```
histograms).
  warnings.warn(msg, FutureWarning)
```

[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ec6713160>



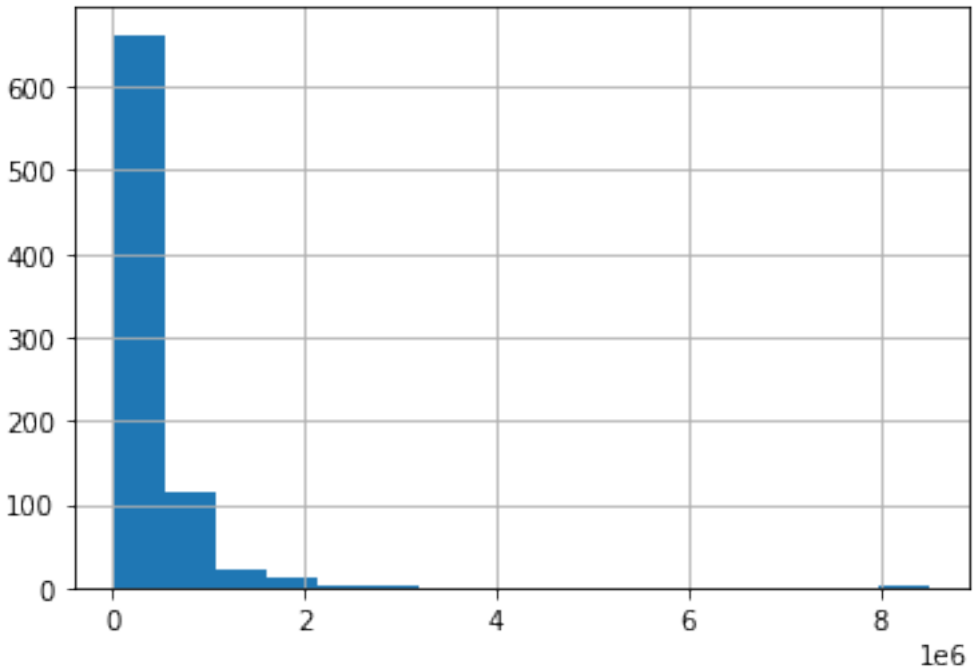
[42]:
```
sns.kdeplot(data=hello['Price'],label="Price",shade=True)
```

[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ec6619100>



[43]:
```
hello['Price'].hist(bins=16)
```

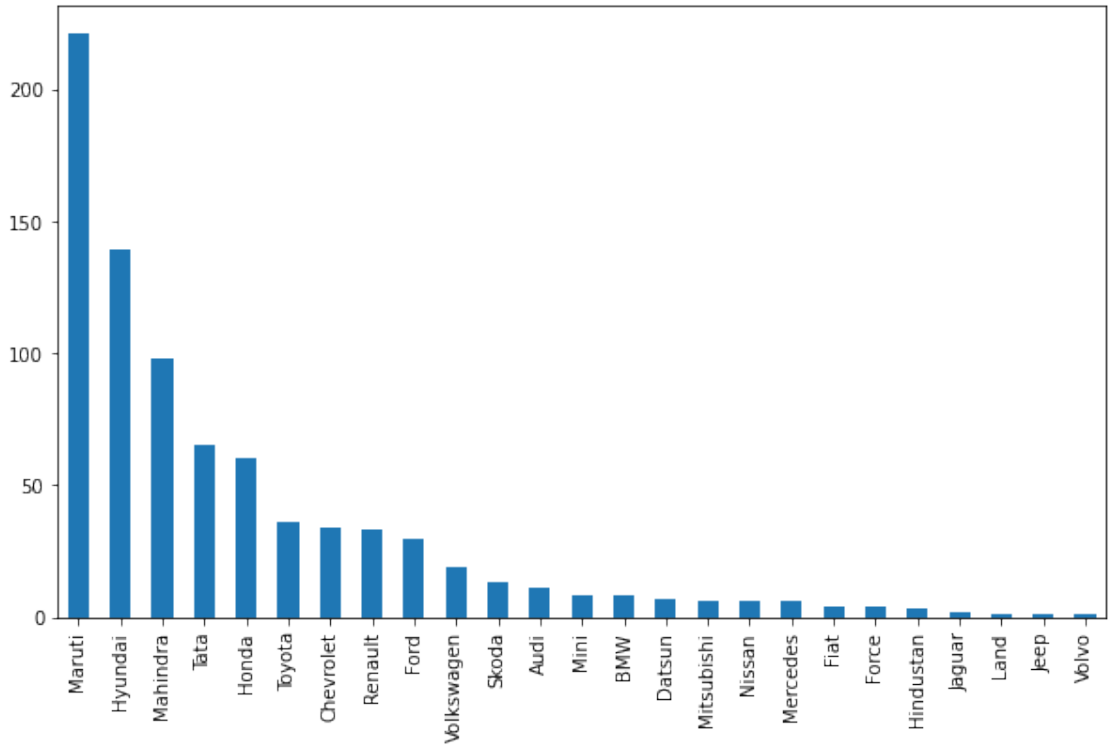
[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ec660f790>



[43]:

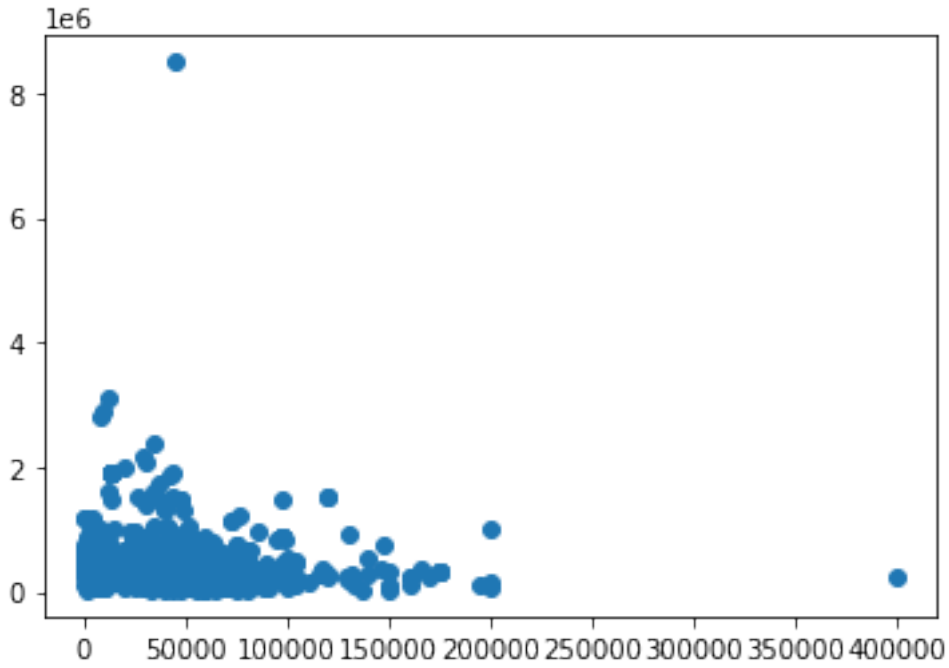
[44]:
```
hello['company'].value_counts().plot(kind='bar',figsize=[10,6])
```

[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ec4031c70>



[45]:
```
plt.scatter(x='kms_driven',y='Price',data=hello)
```

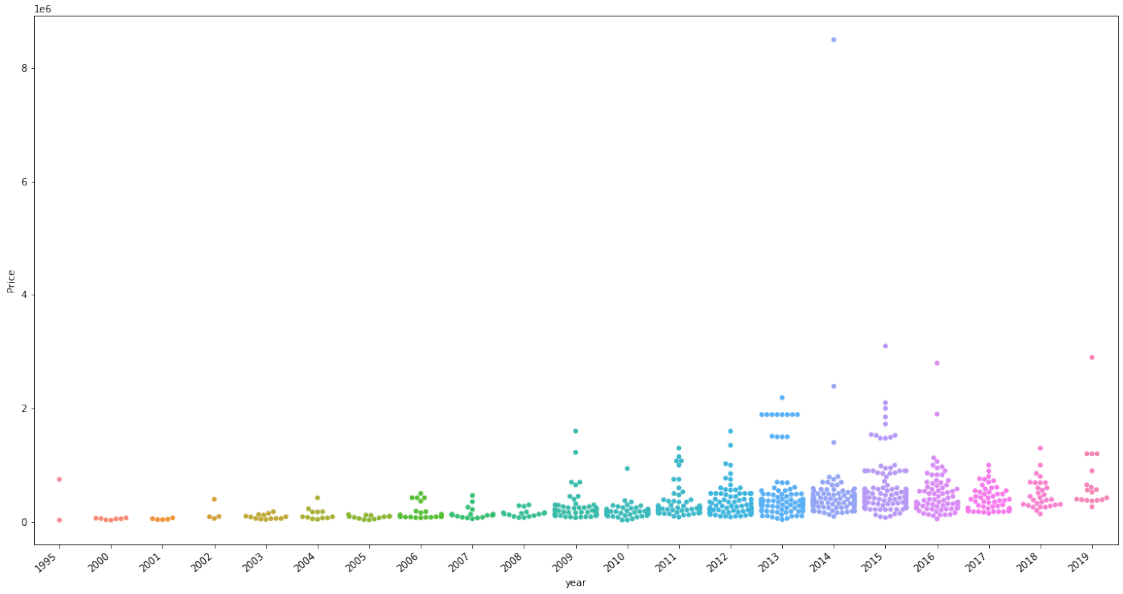
[45]: <matplotlib.collections.PathCollection at 0x7f1ec3fa5580>



[46]:
```
plt.subplots(figsize=(20,10))
ax=sns.swarmplot(x='year',y='Price',data=hello)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```

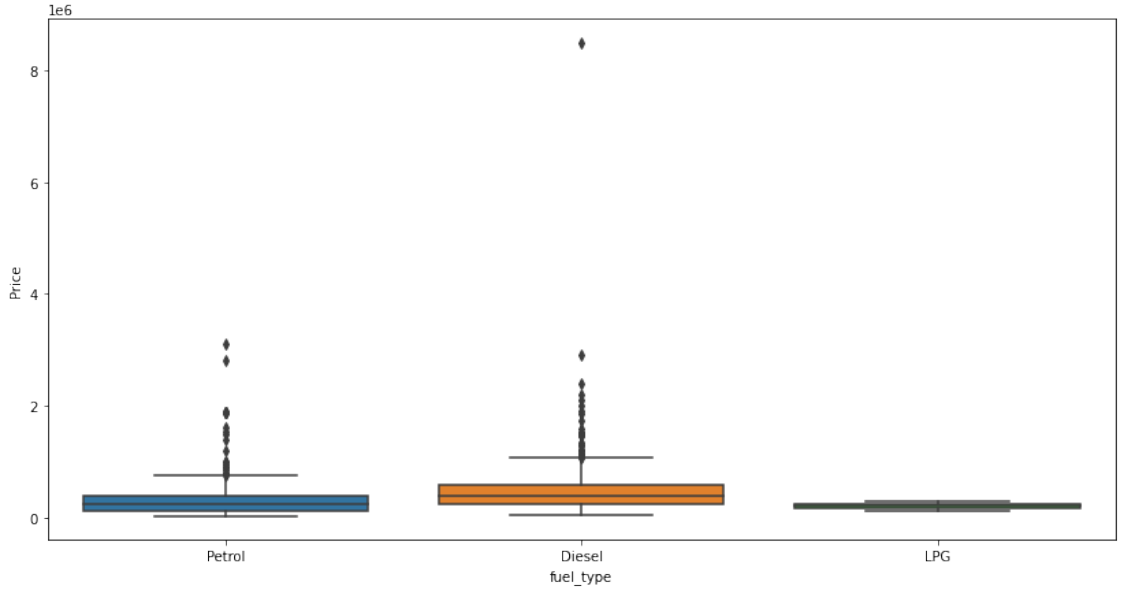
```
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
15.4% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
22.7% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
26.3% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
12.5% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
38.9% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
34.9% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
37.3% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
28.0% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
29.8% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
38.0% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
37.8% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
14.9% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:1296: UserWarning:
17.0% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



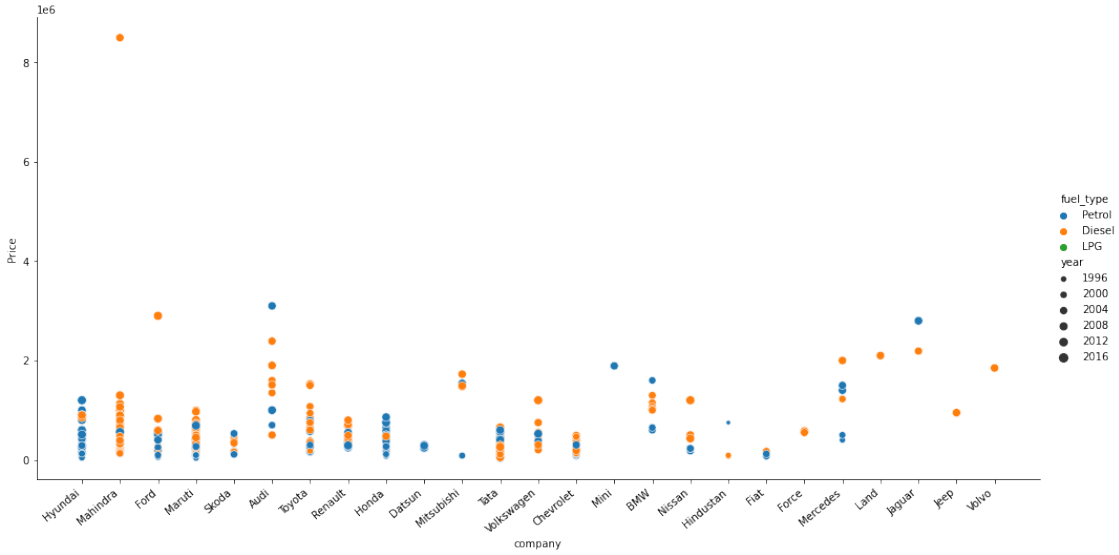
[47]:
```
plt.subplots(figsize=(14,7))
sns.boxplot(x='fuel_type',y='Price',data=hello)
```

[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1ec4027430>
```



[48]:
```
ax=sns.
  ↪relplot(x='company',y='Price',data=hello,hue='fuel_type',size='year',height=7,aspect=2)
ax.set_xticklabels(rotation=40,ha='right')
```

[48]: <seaborn.axisgrid.FacetGrid at 0x7f1ebfd64a60>



[49]:
```
hello['name']
```

[49]:
```
0        Hyundai Santro Xing XO eRLX Euro III
1                     Mahindra Jeep CL550 MDI
3      Hyundai Grand i10 Magna 1.2 Kappa VTVT
4            Ford EcoSport Titanium 1.5L TDCi
6                                   Ford Figo
                        ...
883               Maruti Suzuki Ritz VXI ABS
885                Tata Indica V2 DLE BS III
886                     Toyota Corolla Altis
888                      Tata Zest XM Diesel
889                      Mahindra Quanto C8
Name: name, Length: 816, dtype: object
```

[53]:
```
hello['name']=hello['name'].str.split(' ').str.slice(0,3).str.join('')
hello
```

[53]:

	name	company	year	Price	kms_driven	fuel_type
0	HyundaiSantroXing	Hyundai	2007	80000	45000	Petrol
1	MahindraJeepCL550	Mahindra	2006	425000	40	Diesel
2	HyundaiGrandi10	Hyundai	2014	325000	28000	Petrol
3	FordEcoSportTitanium	Ford	2014	575000	36000	Diesel
4	FordFigo	Ford	2012	175000	41000	Diesel
..	...	...	...	...	...	...
811	MarutiSuzukiRitz	Maruti	2011	270000	50000	Petrol
812	TataIndicaV2	Tata	2009	110000	30000	Diesel

```
813    ToyotaCorollaAltis    Toyota  2009  300000      132000    Petrol
814            TataZestXM      Tata  2018  260000       27000    Diesel
815      MahindraQuantoC8  Mahindra  2013  390000       40000    Diesel

[816 rows x 6 columns]
```

[57]:
```
hello=hello.reset_index(drop=True)
hello
```

[57]:
```
                    name   company  year   Price  kms_driven fuel_type
0       HyundaiSantroXing   Hyundai  2007   80000       45000    Petrol
1       MahindraJeepCL550  Mahindra  2006  425000          40    Diesel
2         HyundaiGrandi10   Hyundai  2014  325000       28000    Petrol
3    FordEcoSportTitanium      Ford  2014  575000       36000    Diesel
4                FordFigo      Ford  2012  175000       41000    Diesel
..                     …       …   …     …          …       …
811      MarutiSuzukiRitz    Maruti  2011  270000       50000    Petrol
812          TataIndicaV2      Tata  2009  110000       30000    Diesel
813    ToyotaCorollaAltis    Toyota  2009  300000      132000    Petrol
814            TataZestXM      Tata  2018  260000       27000    Diesel
815      MahindraQuantoC8  Mahindra  2013  390000       40000    Diesel

[816 rows x 6 columns]
```

[58]:
```
hello.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        816 non-null    object
 1   company     816 non-null    object
 2   year        816 non-null    int64
 3   Price       816 non-null    int64
 4   kms_driven  816 non-null    int64
 5   fuel_type   816 non-null    object
dtypes: int64(3), object(3)
memory usage: 38.4+ KB
```

[59]:
```
hello.describe()
```

[59]:
```
              year         Price     kms_driven
count   816.000000  8.160000e+02     816.000000
mean   2012.444853  4.117176e+05   46275.531863
std       4.002992  4.751844e+05   34297.428044
min    1995.000000  3.000000e+04       0.000000
```

```
25%    2010.000000  1.750000e+05   27000.000000
50%    2013.000000  2.999990e+05   41000.000000
75%    2015.000000  4.912500e+05   56818.500000
max    2019.000000  8.500003e+06  400000.000000
```

[60]:
```
hello[hello['Price']>6e6]
```

[60]:
```
                name   company  year    Price  kms_driven fuel_type
534  MahindraXUV500W6  Mahindra  2014  8500003       45000    Diesel
```

[61]:
```
hello=hello[hello['Price']<6e6].reset_index(drop=True)
hello
```

[61]:
```
                    name   company  year   Price  kms_driven fuel_type
0       HyundaiSantroXing   Hyundai  2007   80000       45000    Petrol
1       MahindraJeepCL550  Mahindra  2006  425000          40    Diesel
2         HyundaiGrandi10   Hyundai  2014  325000       28000    Petrol
3    FordEcoSportTitanium      Ford  2014  575000       36000    Diesel
4                FordFigo      Ford  2012  175000       41000    Diesel
..                    …       …   …     …          …       …
810      MarutiSuzukiRitz    Maruti  2011  270000       50000    Petrol
811          TataIndicaV2      Tata  2009  110000       30000    Diesel
812    ToyotaCorollaAltis    Toyota  2009  300000      132000    Petrol
813            TataZestXM      Tata  2018  260000       27000    Diesel
814      MahindraQuantoC8  Mahindra  2013  390000       40000    Diesel

[815 rows x 6 columns]
```

[62]:
```
hello.shape
```

[62]:
```
(815, 6)
```

[63]:
```
hello.to_csv('cleand car.csv')
```

[64]:
```
X=hello.drop(columns='Price')
```

[65]:
```
X
```

[65]:
```
                    name   company  year  kms_driven fuel_type
0       HyundaiSantroXing   Hyundai  2007       45000    Petrol
1       MahindraJeepCL550  Mahindra  2006          40    Diesel
2         HyundaiGrandi10   Hyundai  2014       28000    Petrol
3    FordEcoSportTitanium      Ford  2014       36000    Diesel
4                FordFigo      Ford  2012       41000    Diesel
..                    …       …   …          …       …
810      MarutiSuzukiRitz    Maruti  2011       50000    Petrol
811          TataIndicaV2      Tata  2009       30000    Diesel
```

```
812    ToyotaCorollaAltis    Toyota  2009      132000    Petrol
813            TataZestXM      Tata  2018       27000    Diesel
814      MahindraQuantoC8  Mahindra  2013       40000    Diesel

[815 rows x 5 columns]
```

[66]:
```
X['company']
```

[66]:
```
0       Hyundai
1      Mahindra
2       Hyundai
3          Ford
4          Ford
         …
810      Maruti
811        Tata
812      Toyota
813        Tata
814    Mahindra
Name: company, Length: 815, dtype: object
```

[68]:
```
Y=hello['Price']
```

[69]:
```
Y
```

[69]:
```
0       80000
1      425000
2      325000
3      575000
4      175000
         …
810    270000
811    110000
812    300000
813    260000
814    390000
Name: Price, Length: 815, dtype: int64
```

[70]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```

[71]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

[72]:
```
ohe=OneHotEncoder()
```

[73]:
```
ohe.fit(X[['name','company','fuel_type']])
```

[73]:
```
OneHotEncoder()
```

[74]:
```
ohe.categories_
```

[74]:
```
[array(['AudiA3Cabriolet', 'AudiA41.8', 'AudiA42.0', 'AudiA62.0', 'AudiA8',
        'AudiQ32.0', 'AudiQ52.0', 'AudiQ7', 'BMW3Series', 'BMW5Series',
        'BMW7Series', 'BMWX1', 'BMWX1sDrive20d', 'BMWX1xDrive20d',
        'ChevroletBeat', 'ChevroletBeatDiesel', 'ChevroletBeatLS',
        'ChevroletBeatLT', 'ChevroletBeatPS', 'ChevroletCruzeLTZ',
        'ChevroletEnjoy', 'ChevroletEnjoy1.4', 'ChevroletSail1.2',
        'ChevroletSailUVA', 'ChevroletSpark', 'ChevroletSpark1.0',
        'ChevroletSparkLS', 'ChevroletSparkLT', 'ChevroletTaveraLS',
        'ChevroletTaveraNeo', 'DatsunGOT', 'DatsunGoPlus', 'DatsunRediGO',
        'FiatLineaEmotion', 'FiatPetraELX', 'FiatPuntoEmotion',
        'ForceMotorsForce', 'ForceMotorsOne', 'FordEcoSport',
        'FordEcoSportAmbiente', 'FordEcoSportTitanium',
        'FordEcoSportTrend', 'FordEndeavor4x4', 'FordFiesta',
        'FordFiestaSXi', 'FordFigo', 'FordFigoDiesel', 'FordFigoDuratorq',
        'FordFigoPetrol', 'FordFusion1.4', 'FordIkon1.3', 'FordIkon1.6',
        'HindustanMotorsAmbassador', 'HondaAccord', 'HondaAmaze',
        'HondaAmaze1.2', 'HondaAmaze1.5', 'HondaBrio', 'HondaBrioV',
        'HondaBrioVX', 'HondaCity', 'HondaCity1.5', 'HondaCitySV',
        'HondaCityVX', 'HondaCityZX', 'HondaJazzS', 'HondaJazzVX',
        'HondaMobilio', 'HondaMobilioS', 'HondaWRV', 'HyundaiAccent',
        'HyundaiAccentExecutive', 'HyundaiAccentGLE', 'HyundaiAccentGLX',
        'HyundaiCreta', 'HyundaiCreta1.6', 'HyundaiElantra1.8',
        'HyundaiElantraSX', 'HyundaiElitei20', 'HyundaiEon', 'HyundaiEonD',
        'HyundaiEonEra', 'HyundaiEonMagna', 'HyundaiEonSportz',
        'HyundaiFluidicVerna', 'HyundaiGetz', 'HyundaiGetzGLE',
        'HyundaiGetzPrime', 'HyundaiGrandi10', 'HyundaiSantro',
        'HyundaiSantroAE', 'HyundaiSantroXing', 'HyundaiSonataTransform',
        'HyundaiVerna', 'HyundaiVerna1.4', 'HyundaiVerna1.6',
        'HyundaiVernaFluidic', 'HyundaiVernaTransform', 'HyundaiVernaVGT',
        'HyundaiXcentBase', 'HyundaiXcentSX', 'Hyundaii10',
        'Hyundaii10Era', 'Hyundaii10Magna', 'Hyundaii10Sportz',
        'Hyundaii20', 'Hyundaii20Active', 'Hyundaii20Asta',
        'Hyundaii20Magna', 'Hyundaii20Select', 'Hyundaii20Sportz',
        'JaguarXEXE', 'JaguarXF2.2', 'JeepWranglerUnlimited',
        'LandRoverFreelander', 'MahindraBoleroDI', 'MahindraBoleroPower',
        'MahindraBoleroSLE', 'MahindraJeepCL550', 'MahindraJeepMM',
        'MahindraKUV100', 'MahindraKUV100K8', 'MahindraLogan',
        'MahindraLoganDiesel', 'MahindraQuantoC4', 'MahindraQuantoC8',
        'MahindraScorpio', 'MahindraScorpio2.6', 'MahindraScorpioLX',
```

```
       'MahindraScorpioS10', 'MahindraScorpioS4', 'MahindraScorpioSLE',
       'MahindraScorpioSLX', 'MahindraScorpioVLX', 'MahindraScorpioVlx',
       'MahindraScorpioW', 'MahindraTUV300T4', 'MahindraTUV300T8',
       'MahindraTharCRDe', 'MahindraXUV500', 'MahindraXUV500W10',
       'MahindraXUV500W6', 'MahindraXUV500W8', 'MahindraXyloD2',
       'MahindraXyloE4', 'MahindraXyloE8', 'MarutiSuzuki800',
       'MarutiSuzukiA', 'MarutiSuzukiAlto', 'MarutiSuzukiBaleno',
       'MarutiSuzukiCelerio', 'MarutiSuzukiCiaz', 'MarutiSuzukiDzire',
       'MarutiSuzukiEeco', 'MarutiSuzukiErtiga', 'MarutiSuzukiEsteem',
       'MarutiSuzukiEstilo', 'MarutiSuzukiMaruti', 'MarutiSuzukiOmni',
       'MarutiSuzukiRitz', 'MarutiSuzukiS', 'MarutiSuzukiSX4',
       'MarutiSuzukiStingray', 'MarutiSuzukiSwift', 'MarutiSuzukiVersa',
       'MarutiSuzukiVitara', 'MarutiSuzukiWagon', 'MarutiSuzukiZen',
       'MercedesBenzA', 'MercedesBenzB', 'MercedesBenzC',
       'MercedesBenzGLA', 'MiniCooperS', 'MitsubishiLancer1.8',
       'MitsubishiPajeroSport', 'NissanMicraXL', 'NissanMicraXV',
       'NissanSunny', 'NissanSunnyXL', 'NissanTerranoXL', 'NissanXTrail',
       'RenaultDuster', 'RenaultDuster110', 'RenaultDuster110PS',
       'RenaultDuster85', 'RenaultDuster85PS', 'RenaultDusterRxL',
       'RenaultKwid', 'RenaultKwid1.0', 'RenaultKwidRXT',
       'RenaultLodgy85', 'RenaultScalaRxL', 'SkodaFabia',
       'SkodaFabia1.2L', 'SkodaFabiaClassic', 'SkodaLaura',
       'SkodaOctaviaClassic', 'SkodaRapidElegance', 'SkodaSuperb1.8',
       'SkodaYetiAmbition', 'TataAriaPleasure', 'TataBoltXM',
       'TataIndica', 'TataIndicaV2', 'TataIndicaeV2', 'TataIndigoCS',
       'TataIndigoLS', 'TataIndigoLX', 'TataIndigoMarina',
       'TataIndigoeCS', 'TataManza', 'TataManzaAqua', 'TataManzaAura',
       'TataManzaELAN', 'TataNano', 'TataNanoCx', 'TataNanoGenX',
       'TataNanoLX', 'TataNanoLx', 'TataSumoGold', 'TataSumoGrande',
       'TataSumoVicta', 'TataTiagoRevotorq', 'TataTiagoRevotron',
       'TataTigorRevotron', 'TataVentureEX', 'TataVistaQuadrajet',
       'TataZestQuadrajet', 'TataZestXE', 'TataZestXM', 'ToyotaCorolla',
       'ToyotaCorollaAltis', 'ToyotaCorollaH2', 'ToyotaEtios',
       'ToyotaEtiosG', 'ToyotaEtiosGD', 'ToyotaEtiosLiva',
       'ToyotaFortuner', 'ToyotaFortuner3.0', 'ToyotaInnova2.0',
       'ToyotaInnova2.5', 'ToyotaQualis', 'VolkswagenJettaComfortline',
       'VolkswagenJettaHighline', 'VolkswagenPassatDiesel',
       'VolkswagenPolo', 'VolkswagenPoloComfortline',
       'VolkswagenPoloHighline', 'VolkswagenPoloHighline1.2L',
       'VolkswagenPoloTrendline', 'VolkswagenVentoComfortline',
       'VolkswagenVentoHighline', 'VolkswagenVentoKonekt',
       'VolvoS80Summum'], dtype=object),
array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford',
       'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
       'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
       'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
      dtype=object),
```

```
       array(['Diesel', 'LPG', 'Petrol'], dtype=object)]
```

[75]:
```
column_trans= make_column_transformer((OneHotEncoder(categories=ohe.
 ↪categories_),['name','company','fuel_type']),remainder='passthrough')
```

[76]:
```
lr=LinearRegression()
```

[77]:
```
pipe=make_pipeline(column_trans,lr)
```

[78]:
```
pipe.fit(X_train,Y_train)
```

[78]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
OneHotEncoder(categories=[array(['AudiA3Cabriolet', 'AudiA41.8', 'AudiA42.0',
'AudiA62.0', 'AudiA8',
       'AudiQ32.0', 'AudiQ52.0', 'AudiQ7', 'BMW3Series', 'BMW5Series',
       'BMW7Series', 'BMWX1', 'BMWX1sDrive20d', 'BMWX1xDrive20d',
       'ChevroletBeat', 'ChevroletBeat…
array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford',
       'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
       'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
       'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
      dtype=object),
array(['Diesel', 'LPG', 'Petrol'], dtype=object)]),
                                                    ['name', 'company',
                                                     'fuel_type'])])),
                ('linearregression', LinearRegression())])
```

[79]:
```
y_pred=pipe.predict(X_test)
```

[80]:
```
y_pred
```

[80]:
```
array([ 249170.16449966,  489350.94909124,  279702.66098148,
        234886.6428657 , -107089.06074843,  215793.63338483,
        611781.83106028,  236261.6689965 ,  354831.43012763,
        371784.97137028,   10309.1584101 ,  -88741.95050661,
        352685.34811521,  684797.82348696,  486893.67895282,
        608189.49649397, 1567145.6195204 ,  238348.88501201,
        225891.49146713,  -62056.46609909,  768737.59799098,
         85077.58052995,  107737.52725897,  447557.48084125,
        263552.4408035 ,  183203.24886979,  395796.50172289,
         18379.28993972,  335211.51629994,  243068.50815471,
        225822.64823411,  466052.80767076,  650545.85709321,
        235525.26667877,  352685.34811521,  106108.43851604,
        268354.72454811,  251813.90590471,  176538.07849299,
        169970.86461252,  263904.19022978,  261771.24158104,
```

```
 238348.88501201,  901700.33777242,  269224.70741277,
 800553.2204078 ,  214772.3866148 ,  112436.21167633,
1109661.53912262,  328520.81702525,  239705.55911852,
 105809.07730219,   12216.38457751,  226571.05126875,
 235920.03637533,  860687.48503444,  349126.07099186,
 684797.82348696,  502229.37145475,  722072.32424561,
  73167.03807415,  351188.54204591,  585486.99198311,
 611332.78923949,  130957.40470651, 1477455.04054658,
1892019.43444004,  276015.24734854,  360465.43560994,
 117691.70180478,  525291.89906254,  549178.39883711,
  51425.30392605,  320462.6109509 ,  698506.49096014,
  43526.39540156,  539548.89980842,  306594.8174837 ,
 295191.63429958,   42261.594273  ,  521555.5165605 ,
 206965.55017644,  500519.14396934,  334857.88348223,
 290999.3608429 ,  684797.82348696,   74092.75224239,
 578434.32889058,  360240.91469954,  229938.86492467,
 149756.91185303,  485779.29327264,  229265.30219349,
 556925.31927598,   74478.29587667,  609221.33071478,
 857594.7797412 ,  241267.65684713,   43002.51327731,
 632444.64504524,  141971.43745661,  632497.91922181,
 540391.38438201, 1229849.19317527,  419804.9631917 ,
 441489.89665525,  258078.10233378,  181856.12340743,
 352179.47027797,  826050.65075391,  511099.12810367,
 347039.8975547 ,  569043.6430537 ,  452528.84141631,
 225719.86190297,  427833.61498474, 1046734.38472909,
 858217.7550201 ,  522534.96227797,  349170.38539685,
 350931.18301082,  675839.22124884,  227811.85505205,
 596842.38512746,  350982.02542253,    3266.78550135,
 860687.48503444,  284145.83390155,  370968.93078124,
 520088.46991025,  455389.69548331,  831253.65133705,
 150355.63428075, 1892019.43444004,  361430.95425389,
 237200.31160712,  201515.84392399,  495237.87255452,
 522609.80258144,  193158.9710317 ,  789156.79377727,
 118421.09685912,  632444.64504524,  183507.33425414,
  81944.60409962,  340356.75573328,  284969.07723966,
 255281.72677851,  486264.49540775,  285476.38874213,
 236261.6689965 ,   34630.31646154,   46186.48268352,
 689790.29149821,  590305.32238134,  230180.70220277,
 502229.37145475,  205674.51542174,  677635.388532  ,
 298995.86853171,   84734.71695455,  131200.41117186,
 282753.96639695])
```

[81]:
```
r2_score(Y_test,y_pred)
```

[81]:
```
0.7020942633301359
```

[82]:
```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.
 ↪1,random_state=i)
    lr=LinearRegression()
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

[83]:
```
np.argmax(scores)
```

[83]: 655

[84]:
```
scores[np.argmax(scores)]
```

[84]: 0.920087093218515

[85]:
```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.
 ↪1,random_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

[85]: 0.920087093218515

[86]:
```
import pickle
```

[87]:
```
pickle.dump(pipe,open('LinearRegressionModel.pkl','wb'))
```

[88]:
```
!pip install pickle--mixi
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
ERROR: Could not find a version that satisfies the requirement pickle--mixi
(from versions: none)
ERROR: No matching distribution found for pickle--mixi
```

[89]:
```
loaded_model=pickle.load(open('LinearRegressionModel.pkl','rb'))
```

[90]:
```
pipe.predict(pd.DataFrame(columns=X_test.columns,data=np.
 ↪array(['MarutiSuzukiSwift','Maruti',2019,100,'Petrol']).reshape(1,5)))
```

[90]: array([416107.38610967])

[91]:
```
pipe.steps[0][1].transformers[0][1].categories[0]
```

[91]:
```
array(['AudiA3Cabriolet', 'AudiA41.8', 'AudiA42.0', 'AudiA62.0', 'AudiA8',
       'AudiQ32.0', 'AudiQ52.0', 'AudiQ7', 'BMW3Series', 'BMW5Series',
       'BMW7Series', 'BMWX1', 'BMWX1sDrive20d', 'BMWX1xDrive20d',
       'ChevroletBeat', 'ChevroletBeatDiesel', 'ChevroletBeatLS',
       'ChevroletBeatLT', 'ChevroletBeatPS', 'ChevroletCruzeLTZ',
       'ChevroletEnjoy', 'ChevroletEnjoy1.4', 'ChevroletSail1.2',
       'ChevroletSailUVA', 'ChevroletSpark', 'ChevroletSpark1.0',
       'ChevroletSparkLS', 'ChevroletSparkLT', 'ChevroletTaveraLS',
       'ChevroletTaveraNeo', 'DatsunGOT', 'DatsunGoPlus', 'DatsunRediGO',
       'FiatLineaEmotion', 'FiatPetraELX', 'FiatPuntoEmotion',
       'ForceMotorsForce', 'ForceMotorsOne', 'FordEcoSport',
       'FordEcoSportAmbiente', 'FordEcoSportTitanium',
       'FordEcoSportTrend', 'FordEndeavor4x4', 'FordFiesta',
       'FordFiestaSXi', 'FordFigo', 'FordFigoDiesel', 'FordFigoDuratorq',
       'FordFigoPetrol', 'FordFusion1.4', 'FordIkon1.3', 'FordIkon1.6',
       'HindustanMotorsAmbassador', 'HondaAccord', 'HondaAmaze',
       'HondaAmaze1.2', 'HondaAmaze1.5', 'HondaBrio', 'HondaBrioV',
       'HondaBrioVX', 'HondaCity', 'HondaCity1.5', 'HondaCitySV',
       'HondaCityVX', 'HondaCityZX', 'HondaJazzS', 'HondaJazzVX',
       'HondaMobilio', 'HondaMobilioS', 'HondaWRV', 'HyundaiAccent',
       'HyundaiAccentExecutive', 'HyundaiAccentGLE', 'HyundaiAccentGLX',
       'HyundaiCreta', 'HyundaiCreta1.6', 'HyundaiElantra1.8',
       'HyundaiElantraSX', 'HyundaiElitei20', 'HyundaiEon', 'HyundaiEonD',
       'HyundaiEonEra', 'HyundaiEonMagna', 'HyundaiEonSportz',
       'HyundaiFluidicVerna', 'HyundaiGetz', 'HyundaiGetzGLE',
       'HyundaiGetzPrime', 'HyundaiGrandi10', 'HyundaiSantro',
       'HyundaiSantroAE', 'HyundaiSantroXing', 'HyundaiSonataTransform',
       'HyundaiVerna', 'HyundaiVerna1.4', 'HyundaiVerna1.6',
       'HyundaiVernaFluidic', 'HyundaiVernaTransform', 'HyundaiVernaVGT',
       'HyundaiXcentBase', 'HyundaiXcentSX', 'Hyundaii10',
       'Hyundaii10Era', 'Hyundaii10Magna', 'Hyundaii10Sportz',
       'Hyundaii20', 'Hyundaii20Active', 'Hyundaii20Asta',
       'Hyundaii20Magna', 'Hyundaii20Select', 'Hyundaii20Sportz',
       'JaguarXEXE', 'JaguarXF2.2', 'JeepWranglerUnlimited',
       'LandRoverFreelander', 'MahindraBoleroDI', 'MahindraBoleroPower',
       'MahindraBoleroSLE', 'MahindraJeepCL550', 'MahindraJeepMM',
       'MahindraKUV100', 'MahindraKUV100K8', 'MahindraLogan',
       'MahindraLoganDiesel', 'MahindraQuantoC4', 'MahindraQuantoC8',
       'MahindraScorpio', 'MahindraScorpio2.6', 'MahindraScorpioLX',
       'MahindraScorpioS10', 'MahindraScorpioS4', 'MahindraScorpioSLE',
       'MahindraScorpioSLX', 'MahindraScorpioVLX', 'MahindraScorpioVlx',
       'MahindraScorpioW', 'MahindraTUV300T4', 'MahindraTUV300T8',
       'MahindraTharCRDe', 'MahindraXUV500', 'MahindraXUV500W10',
```

'MahindraXUV500W6', 'MahindraXUV500W8', 'MahindraXyloD2',
'MahindraXyloE4', 'MahindraXyloE8', 'MarutiSuzuki800',
'MarutiSuzukiA', 'MarutiSuzukiAlto', 'MarutiSuzukiBaleno',
'MarutiSuzukiCelerio', 'MarutiSuzukiCiaz', 'MarutiSuzukiDzire',
'MarutiSuzukiEeco', 'MarutiSuzukiErtiga', 'MarutiSuzukiEsteem',
'MarutiSuzukiEstilo', 'MarutiSuzukiMaruti', 'MarutiSuzukiOmni',
'MarutiSuzukiRitz', 'MarutiSuzukiS', 'MarutiSuzukiSX4',
'MarutiSuzukiStingray', 'MarutiSuzukiSwift', 'MarutiSuzukiVersa',
'MarutiSuzukiVitara', 'MarutiSuzukiWagon', 'MarutiSuzukiZen',
'MercedesBenzA', 'MercedesBenzB', 'MercedesBenzC',
'MercedesBenzGLA', 'MiniCooperS', 'MitsubishiLancer1.8',
'MitsubishiPajeroSport', 'NissanMicraXL', 'NissanMicraXV',
'NissanSunny', 'NissanSunnyXL', 'NissanTerranoXL', 'NissanXTrail',
'RenaultDuster', 'RenaultDuster110', 'RenaultDuster110PS',
'RenaultDuster85', 'RenaultDuster85PS', 'RenaultDusterRxL',
'RenaultKwid', 'RenaultKwid1.0', 'RenaultKwidRXT',
'RenaultLodgy85', 'RenaultScalaRxL', 'SkodaFabia',
'SkodaFabia1.2L', 'SkodaFabiaClassic', 'SkodaLaura',
'SkodaOctaviaClassic', 'SkodaRapidElegance', 'SkodaSuperb1.8',
'SkodaYetiAmbition', 'TataAriaPleasure', 'TataBoltXM',
'TataIndica', 'TataIndicaV2', 'TataIndicaeV2', 'TataIndigoCS',
'TataIndigoLS', 'TataIndigoLX', 'TataIndigoMarina',
'TataIndigoeCS', 'TataManza', 'TataManzaAqua', 'TataManzaAura',
'TataManzaELAN', 'TataNano', 'TataNanoCx', 'TataNanoGenX',
'TataNanoLX', 'TataNanoLx', 'TataSumoGold', 'TataSumoGrande',
'TataSumoVicta', 'TataTiagoRevotorq', 'TataTiagoRevotron',
'TataTigorRevotron', 'TataVentureEX', 'TataVistaQuadrajet',
'TataZestQuadrajet', 'TataZestXE', 'TataZestXM', 'ToyotaCorolla',
'ToyotaCorollaAltis', 'ToyotaCorollaH2', 'ToyotaEtios',
'ToyotaEtiosG', 'ToyotaEtiosGD', 'ToyotaEtiosLiva',
'ToyotaFortuner', 'ToyotaFortuner3.0', 'ToyotaInnova2.0',
'ToyotaInnova2.5', 'ToyotaQualis', 'VolkswagenJettaComfortline',
'VolkswagenJettaHighline', 'VolkswagenPassatDiesel',
'VolkswagenPolo', 'VolkswagenPoloComfortline',
'VolkswagenPoloHighline', 'VolkswagenPoloHighline1.2L',
'VolkswagenPoloTrendline', 'VolkswagenVentoComfortline',
'VolkswagenVentoHighline', 'VolkswagenVentoKonekt',
'VolvoS80Summum'], dtype=object)