In [1]:
```
import pandas as pd
import numpy as np
import pickle
#import warnings
#warnings.filterwarnings("ignore")
```

In [2]:
```
data = pd.read_csv('carIndia.csv')
```

In [3]:
```
data
```

Out[3]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
0	2012	Maruti	Alto K10 VXI	Mumbai	29067	Petrol	2nd Owner	165199
1	2011	Hyundai	i20 SPORTZ 1.2 O	Mumbai	36791	Petrol	2nd Owner	326099
2	2010	Maruti	A Star VXI	Mumbai	35171	Petrol	1st Owner	195199
3	2011	Hyundai	Santro Xing GLS	Mumbai	19908	Petrol	1st Owner	195199
4	2012	Hyundai	Santro Xing GLS	Mumbai	43847	Petrol	3rd Owner	203299
...	...	...	...	...	...	...	...	...
3360	2014	Honda	City S MT DIESEL	Kolkata	61643	Diesel	3rd Owner	500000
3361	2006	Maruti	Wagon R LXI	Kolkata	26500	Petrol	1st Owner	100000
3362	2016	Maruti	S Cross ZETA 1.3	Kolkata	57828	Diesel	1st Owner	550000
3363	2012	BMW	3 Series 320D	Kolkata	23782	Diesel	2nd Owner	1200000
3364	2016	Hyundai	Creta 1.4 BASE	Kolkata	33130	Diesel	1st Owner	850000

3365 rows × 8 columns

In [4]:
```
data.head(10)
```

Out[4]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
0	2012	Maruti	Alto K10 VXI	Mumbai	29067	Petrol	2nd Owner	165199
1	2011	Hyundai	i20 SPORTZ 1.2 O	Mumbai	36791	Petrol	2nd Owner	326099
2	2010	Maruti	A Star VXI	Mumbai	35171	Petrol	1st Owner	195199
3	2011	Hyundai	Santro Xing GLS	Mumbai	19908	Petrol	1st Owner	195199
4	2012	Hyundai	Santro Xing GLS	Mumbai	43847	Petrol	3rd Owner	203299
5	2016	Hyundai	Eon SPORTZ	Mumbai	21303	Petrol	1st Owner	291299
6	2010	Maruti	Alto K10 VXI	Mumbai	50742	Petrol	2nd Owner	170399
7	2015	Maruti	Alto K10 VXI	Mumbai	12657	Petrol	1st Owner	282299
8	2013	Maruti	Wagon R 1.0 LXI	Mumbai	13688	Petrol	1st Owner	326199
9	2014	Hyundai	i10 MAGNA 1.1 IRDE2	Mumbai	13068	Petrol	1st Owner	369699

In [5]:
```
data.tail(10)
```

Out[5]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
3355	2017	Renault	Kwid RXT	Kolkata	36603	Petrol	1st Owner	290799
3356	2014	Hyundai	Eon D LITE PLUS	Kolkata	62278	Petrol	1st Owner	231399
3357	2001	Maruti	Wagon R LXI	Kolkata	72000	Petrol	2nd Owner	38000
3358	2010	Hyundai	i20 MAGNA O 1.2	Kolkata	54757	Petrol	2nd Owner	160000
3359	2010	Honda	City S MT PETROL	Kolkata	70410	Petrol	3rd Owner	225000
3360	2014	Honda	City S MT DIESEL	Kolkata	61643	Diesel	3rd Owner	500000
3361	2006	Maruti	Wagon R LXI	Kolkata	26500	Petrol	1st Owner	100000
3362	2016	Maruti	S Cross ZETA 1.3	Kolkata	57828	Diesel	1st Owner	550000
3363	2012	BMW	3 Series 320D	Kolkata	23782	Diesel	2nd Owner	1200000
3364	2016	Hyundai	Creta 1.4 BASE	Kolkata	33130	Diesel	1st Owner	850000

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3365 entries, 0 to 3364
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   model_year             3365 non-null   int64 
 1   maker                  3365 non-null   object
 2   model_name             3365 non-null   object
 3   city                   3365 non-null   object
 4   distance_covered (km)  3365 non-null   int64 
 5   fuel_type              3365 non-null   object
 6   pre_owner              3365 non-null   object
 7   price (₹)              3365 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 210.4+ KB
```

In [7]:
```
list(data)
```

Out[7]:
```
['model_year',
 'maker',
 'model_name',
 'city',
 'distance_covered (km)',
 'fuel_type',
 'pre_owner',
 'price (₹)']
```

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

Out[8]:
```
model_year             0
maker                  0
model_name             0
city                   0
distance_covered (km)  0
fuel_type              0
pre_owner              0
price (₹)              0
dtype: int64
```

In [9]:
```
data['model_name'].unique()
```

Out[9]:
```
array(['Alto K10 VXI', 'i20 SPORTZ 1.2 O', 'A Star VXI',
       'Santro Xing GLS', 'Eon SPORTZ', 'Wagon R 1.0 LXI',
       'i10 MAGNA 1.1 IRDE2', 'Swift Dzire VXI', 'Brio 1.2 S MT I VTEC',
       'Swift Dzire VXI 1.2 BS IV', 'Etios Liva G', 'Wagon R 1.0 VXI',
       'Alto LXI', 'Zen Estilo LXI', 'i10 MAGNA 1.2 KAPPA2',
       'Wagon R 1.0 LXI CNG', 'Figo 1.2 ZXI DURATEC',
       'Grand i10 SPORTZ 1.2 KAPPA VTVT', 'Swift ZXI',
       'Jetta TRENDLINE 1.6', 'i10 SPORTZ 1.1 IRDE2', 'Etios G',
       'Celerio ZXI AMT', 'Alto 800 VXI', 'Swift VXI', 'Swift VDI',
       'Grand i10 SPORTS 1.2 VTVT', 'i10 SPORTZ 1.2 KAPPA2',
       'Amaze 1.2 SMT I VTEC', 'Ritz VXI BS IV', 'Alto 800 LXI',
       'i10 SPORTZ 1.2 AT KAPPA2', 'i20 ASTA 1.2', 'Alto K10 LXI',
       'Celerio VXI', 'Celerio VXI AMT', 'Alto K10 VXI AMT',
       'SELTOS GTX + AT PETROL', 'Grand i10 ASTA 1.2 KAPPA VTVT',
       'Kuv100 K6+ 6 STR', 'Wagon R VXI', 'Celerio ZXI OPT',
       'Grand i10 ASTA 1.2 (O) VTVT', 'i20 MAGNA 1.2 VTVT',
       'i20 MAGNA O 1.2', 'Baleno DELTA 1.2 K12', 'Baleno ALPHA 1.2 K12',
       'Polo GT TSI 1.2 PETROL AT', 'Polo COMFORTLINE 1.2L PETROL',
       'Swift Dzire LDI BS IV', 'Polo HIGHLINE1.2L PETROL',
       'Verna FLUIDIC 1.6 SX VTVT OPT AT', 'NEW SANTRO 1.1 SPORTS AMT',
       'Go Plus T', 'i10 MAGNA 1.2', 'i20 Active 1.2 S',
       'Benz CLS Class 350 BLUE EFFICIENCY', 'Etios V',
       'i20 SPORTZ 1.2 VTVT', 'Elite i20 SPORTZ 1.2', 'Swift LXI',
       'TIGOR Revotron XT', 'City S MT PETROL',
       'Elite i20 SPORTZ (O) 1.2', 'Elite i20 ASTA 1.2', 'Jazz 1.2 V AT',
       'Vento HIGHLINE TDI AT', 'Ertiga ZDI', 'Swift Dzire VDI BS IV',
       'Celerio ZXI', 'City VX MT PETROL', 'Kwid RXT Opt',
       'Amaze 1.5 EXMT I DTEC', 'Baleno ALPHA DDIS 190',
       'Redi Go 1.0 S AT', 'Figo Aspire 1.2 Trend+ Petrol',
       'IGNIS ZETA 1.2 K12 AMT', 'Baleno DELTA 1.2 K12 AMT',
       'Verna FLUIDIC 1.6 SX VTVT OPT', 'WR-V 1.2 i-VTEC VX MT',
       'TUV300 T8', 'Vitara Brezza VDI OPT', 'Innova 2.5 GX 8 STR BS IV',
       'Amaze 1.2 EXMT I VTEC', 'Ertiga VXI OPTIONAL', 'Ertiga VDI SHVS',
       'Rapid AMBITION 1.6 MPI MT PLUS', 'Dzire VDI',
       'Swift Dzire VXI AMT', 'Grand i10 SPORTZ (O) 1.2 AT VTVT',
       'Grand i10 ASTA 1.2 KAPPA VTVT OPT', 'Creta 1.6 E + VTVT',
       'VENUE 1.0L Turbo GDI SX MT', 'Vitara Brezza VDI', 'XUV500 W6 4X2',
       'XUV500 W8 FWD', 'City V MT PETROL', 'SELTOS GTX+ 1.4 MT',
       'Baleno ZETA 1.2 K12 CVT', 'Polo HIGHLINE1.2L DIESEL',
       'Innova 2.5 G4 8 STR', 'Sunny XL DIESEL', 'Verna FLUIDIC 1.4 VTVT',
       'Corolla Altis 1.8 G', 'Terrano XL PLUS 85 PS DEISEL',
       'Ertiga ZDI PLUS SHVS', 'SELTOS HTX+ MT 1.5 DIESEL',
       'i20 ASTA 1.4 CRDI', 'Verna 1.6 SX VTVT AT (O)', 'XUV500 W3',
       'Swift ZDI', 'Ritz VDI', 'Ritz VXI',
       'Ertiga VDI SHVS LIMITED EDITION',
       'Benz C Class 200 CGI AVANTGARDE', 'Fortuner 2.8 4x2 MT',
       'Vitara Brezza ZDI', 'NEXON XZA + 1.2 PETROL A/T',
       'Verna 1.4 VTVT EX', 'Swift Dzire ZXI 1.2 BS IV',
       'Tiago XT 1.05 REVOTORQ', 'Verna 1.6 SX (O) CRDI MT',
       'Vento HIGHLINE DIESEL', 'Benz C Class C200 CGI GRAND EDITION',
       'Compass 2.0 LIMITED', 'Ciaz ZETA 1.4 VVT AMT',
       'Baleno RS 1.0 PETROL', 'VENUE 1.0 TURBO GDI SX+ AT',
       'Scorpio S11', 'Vitara Brezza ZDI+ DUAL TONE AMT', 'Ciaz ZXI AMT',
       'Duster 85 PS RXE', 'Ritz ZXI', 'Swift Dzire VDI',
       'Superb 1.8 TSI STYLE AT', 'City SV MT DIESEL', 'Polo GT TSI',
       'Accord 2.4 MT', 'Grand i10 ASTA 1.2 AT VTVT',
       'Vitara Brezza ZDI + AMT', 'Benz C Class C220 CDI GRAND EDITION',
       'HECTOR SHARP 2.0 DIESEL', 'Ecosport 1.5TITANIUM TDCI',
       'Innova 2.5 V 8 STR', 'Corolla Altis D 4D GL',
       'Innova 2.5 VX 8 STR BS IV', 'S Cross DELTA 1.3',
       'Figo 1.4 TITANIUM DURATORQ', 'Ritz LDI',
       'Superb 2.0 TDI CR LK AT', 'Corolla Altis VL AT',
       'Grand i10 SPORTZ 1.1 CRDI', 'Yeti ACTIVE 2.0 TDI 4X2',
       'New Elantra 1.6 SX AT O', 'Swift Dzire ZDI', '5 Series 520D 2.0',
       'Fortuner 3.0 AT 4X2', 'Innova 2.5 VX 7 STR BS IV',
       'Vento HIGHLINE 1.2 TSI AT', 'Baleno DELTA DDIS 190', 'Ciaz VDI',
       'Compass LIMITED 1.4 AT', 'Benz E Class E 200 AVANTGARDE',
       'Prius 1.8 Z3', 'Eon ERA PLUS', 'Tiago XZA 1.2 REVOTRON',
       'Kwid RXT 1.0 EASY-R AT OPTION', 'Etios CROSS 1.2 G',
       'Jazz 1.2 SELECT I VTEC', 'i10 ERA 1.1 IRDE',
       'Kwid CLIMBER 1.0 AT', 'Jazz 1.2 BASE I VTEC',
       'Eeco 5 STR CNG WITH AC PLUSHTR', 'Elite i20 ASTA 1.2 (O)',
       'Ameo HIGHLINE 1.2', 'Verna 1.6 SX VTVT', 'Swift VXI OPT',
       'Elite i20 Magna Executive 1.2', 'HECTOR SHARP DCT PETROL',
       'Quanto C4', 'Xcent S 1.2', 'Figo Aspire 1.5 TREND DIESEL',
       'Wagon R LXI', 'Ertiga VDI ABS', 'Zen Estilo VXI', 'Dzire VXI',
       'Alto 800 LXI CNG', 'Manza VX QUADRAJET', 'Civic 1.8V MT',
       'i20 MAGNA O 1.4 CRDI', 'Kwid 1.0 RXT', 'Ciaz ZETA 1.4 VVT',
       'Accord 2.4 AT I VTEC', 'i20 Active 1.2 SX', 'Corolla Altis G',
       'Civic ZX CVT PETROL', 'City S AT', 'i10 ASTA 1.2 KAPPA2',
       'TUV300 T8 AT', 'Elite i20 ASTA 1.4 CRDI',
       'Benz E Class E 250 CDI AVANTGARDE', 'Baleno ZETA 1.2 K12',
       'Kwid RXT', 'Amaze 1.5 EMT I DTEC', 'Ciaz ZDI+ SHVS', 'Hexa XM+',
       'Amaze 1.5 SMT I DTEC', '3 Series 320D SPORTLINE',
       'S Cross ZETA 1.3', 'Sunny XV DIESEL', 'XUV500 W10 AT FWD',
       'City VX MT DIESEL', 'Innova 2.5 G4 7 STR',
       'Amaze 1.2 VXMT I VTEC', 'Rapid 1.6 TDI MT AMBITION PLUS',
       'Santa Fe 2WD AT', 'Ecosport 1.0 ECOBOOST TITANIUM OPT',
       'Benz E Class E 250 CDI ELEGANCE', 'Scorpio S10 AT',
       'Innova Crysta 2.4 VX 8 STR', 'Mobilio 1.5 V I DTEC',
       'Creta 1.6 SX CRDI', 'Scorpio S10', 'Ciaz ZDI',
       'Fortuner 2.8 4x2 AT', 'Xcent SX 1.2 OPT', 'Ritz LXI',
       'Redi Go T (O)', 'Beat LS PETROL', 'Brio 1.2 V MT I VTEC',
       'Wagon R Stingray VXI',
       'Wagon R 1.0 LXI CNG AVANCE LIMITED EDITION', 'Eon MAGNA PLUS',
       'Swift Dzire VXI 1.3', 'Alto LXI CNG',
       'Verna FLUIDIC 1.6 EX VTVT AT', 'Jazz 1.2 S MT',
       'Superb ELEGANCE 1.8 TSI MT', 'Ertiga VXI', 'Eon D LITE PLUS',
       'Swift Dzire VDI ABS', 'Alto VXI', 'Alto K10 VXI (O) AMT',
       'Ritz ZXI ABS', 'Grand i10 MAGNA 1.2 VTVT', 'A Star LXI',
       'Wagon R 1.0 VXI AMT', 'Swift LDI BS IV', 'A Star VXI ABS AT',
       'Terrano XL 85 PS DEISEL', 'Ciaz VDI+ SHVS', 'Xcent S 1.1 CRDI',
       'Wagon R LXI MINOR', 'Swift VDI ABS', 'Alto K10 VXI OPT',
       'Etios Liva GD', 'Grand i10 MAGNA 1.2 KAPPA VTVT',
       'Duster RXL DIESEL 110', 'Verna FLUIDIC 1.6 SX CRDI',
       'Laura AMBIENTE 1.8 TSI', 'Swift LXI 1.3', 'Duster 85 PS RXL',
       'Polo COMFORTLINE 1.2L DIESEL', 'Tiago XZ 1.05 REVOTORQ',
       'Amaze 1.2 SAT I VTEC', 'Beat LS DIESEL', 'Etios Liva D 4D GD',
       'Creta 1.6 S', 'Rexton RX7', 'i20 SPORTZ 1.4 CRDI',
       'City 1.5 E MT PETROL', 'Etios GD', 'City V CVT',
       'Innova Crysta Touring Sport Diesel MT',
       'Elite i20 MAGNA 1.4 CRDI', 'Santro Xing GL',
       'Creta 1.6 SX (O) CRDI', 'Superb ELEGANCE 1.8 TSI AT',
       'Corolla H2 1.8 E', 'Ecosport 1.5 AMBIENTE TDCI',
       'VENUE SX(O) CRDi', 'Creta 1.6 SX AT PETROL',
       'Duster RXZ AMT 110 PS', 'Vento HIGHLINE PETROL AT',
       'S Cross ALPHA 1.3', 'City S MT DIESEL',
       'Grand i10 MAGNA 1.1 CRDI', 'Jetta TRENDLINE 1.4 TSI MT',
       'Duster 85 PS RXL OPT', 'S Cross ALPHA SHVS',
       'Punto EVO MULTIJET 1.3 90 HP', 'i10 ERA', 'Dzire VDI AMT',
       'Innova 2.5 GX 7 STR BS IV', 'Etios Liva GD exclusive',
       'Benz E Class E 220 CDI ELEGANCE', 'Swift Dzire VXI AT',
       'Cruze LTZ', 'Camry HYBRID', 'Vitara Brezza ZDI PLUS',
       'Etios Liva D 4D GD SP', 'Jetta HIGHLINE TDI AT',
       'Verna FLUIDIC 1.6 EX CRDI', 'Ciaz DELTA 1.3 DDIS SHVS',
       'CRV 2.0 MT', 'Fortuner 3.0 MT 4X2', 'XUV500 W10',
       'Duster RXZ DIESEL 110', '3 Series 320D LUXURYLINE',
       'Creta 1.4 S PLUS CRDI', 'Fortuner 3.0 MT 4X4',
```

'Vitara Brezza ZXI AT SHVS', 'Eon ERA PLUS (O)', 'S PRESSO VXI',
'Xcent E PLUS', 'Kuv100 K2 6 STR', 'i20 ERA 1.2', 'Alto LX',
'Sunny XL PETROL', 'New Wagon-R LXI 1.0 L', 'Nano XT TWIST',
'Swift LXI OPT', 'Ecosport 1.5AMBIENTE TI VCT', 'Dzire LXI',
'TRIBER 1.0 RXL PETROL', 'Kwid RXL', 'Eeco 5 STR CNG WITH HTR',
'Xcent S 1.2 OPT', 'City SV MT PETROL', 'Creta 1.4 S CRDI',
'Ciaz ALPHA 1.5 AT VTVT SHVS', 'New Wagon-R LXI CNG 1.0 L',
'Grand i10 1.2 ASTA (O) AT', 'IGNIS SIGMA 1.2 K12',
'i10 SPORTZ 1.2', 'Fiesta 1.6 EXI LTD', 'Ciaz DELTA 1.4 VVT AT',
'Creta 1.4 E PLUS CRDI', 'Baleno SIGMA 1.2 K12',
'Benz E Class E 220 CDI CLASSIC', 'YARIS V MT',
'Eeco 5 STR WITH AC PLUSHTR', 'Santro Xing GL PLUS',
'GRAND I10 NIOS SPORTZ 1.2 AT', 'Grand Punto ACTIVE 1.2',
'Kwid RXT 1.0 EASY-R AT', 'Spark LS 1.0', 'Nano TWIST XTA',
'Eeco 5 STR WITH AC PLUS HTR',
'Swift Dzire VXI Regal Limited edition', 'OMNI E 8 STR',
'Swift Dzire LXI 1.2 BS IV', 'TRIBER 1.0 RXZ', 'Beat LT PETROL',
'Spark LS 1.0 LPG', 'Micra XL PETROL',
'Grand i10 ASTA AT 1.2 KAPPA VTVT', 'i10 ASTA 1.2 WITH SUNROOF',
'Alto XCITE', 'i20 ASTA 1.2 O WITH SUNROOF',
'Tiago XZ 1.2 REVOTRON', 'Santro Xing XO ERLX EURO III',
'Wagon R VXI BS III', 'Swift VXI ABS', 'i10 MAGNA 1.2 AT',
'Rapid 1.6 TDI MT AMBITION', 'Spark LT 1.0', 'E2O T2',
'YARIS J MT', 'Eon ERA', 'Wagon R VXI MINOR',
'Duster RXL PETROL 104', 'Celerio VXI (O)',
'i20 SPORTZ O 1.4 CRDI', 'Swift ZXI 1.3', 'i20 Active 1.4 SX',
'i10 SPORTZ 1.2 AT', 'Swift VXI 1.3',
'Verna FLUIDIC 1.6 CRDI SX AT', 'i10 MAGNA', 'Ciaz VXI PLUS',
'Corolla Altis D 4D G', 'Elite i20 ASTA 1.2 DUAL TONE',
'Indica Vista VX QUADRAJET', 'Polo GT TDI 1.6 MT DIESEL',
'IGNIS ZETA 1.2 K12', 'Fiesta 1.6 ZXI',
'Fiesta Classic 1.4 CLXI TDCI', 'A3 35TDI',
'Baleno ZETA 1.2 K12 AMT', 'Eon MAGNA', 'Omni 5 SEATER',
'Nano LX SPECIAL EDITION', 'Alto 800 LXI OPT',
'Tiago XE 1.2 REVOTRON', 'Celerio ZXI OPT AMT',
'Wagon R Duo LXI LPG', 'Figo 1.4 EXI DURATORQ', 'OMNI E STD',
'Verna FLUIDIC 1.6 SX VTVT', 'Pulse RX L PETROL',
'Figo 1.2 TITANIUM DURATEC', 'Eon MAGNA PLUS BLUE DRIVE',
'Santro Xing GLS LPG', 'Micra XE DIESEL', 'Tiago XT 1.2 REVOTRON',
'Polo Trendline 1.0 L Petrol', 'Thar CRDE 4X4 BS IV',
'S PRESSO VXI PLUS', 'Swift Dzire VDI OPT', 'Wagon R 1.0 LXI LPG',
'Swift LDI', 'Creta 1.6 SX PLUS AUTO PETROL', 'City VX CVT PETROL',
'Innova Crysta 2.4 GX 8 STR', 'City V AT', 'Figo 1.4 ZXI DURATORQ',
'Baleno ZETA DDIS 190', 'Ecosport 1.5 TREND+ TDCI',
'i10 MAGNA 1.1 LPG', 'Swift VXI AMT', 'Swift ZXI AMT',
'City 1.5 V MT', 'Elite i20 MAGNA 1.2', 'Cruze LTZ AT',
'Ritz VXI GENUS', 'Beat PS DIESEL', 'Eeco 7 STR',
'Grand i10 ASTA 1.2 VTVT', 'Grand i10 SPORTZ O 1.2',
'Indigo CS GLX', 'Tiago XZ+ 1.2 Revotron', 'BR-V 1.5 i-VTEC V CVT',
'Ecosport 1.5 TITANIUM TI VCT AT', 'Elite i20 1.4 CRDI ASTA (O)',
'Swift ZXI+ BS IV', 'Wagon R 1.0 VXI ABS AIR BAG',
'TIGOR XM REVOTORQ', 'Rapid STYLE 1.6 MPI MT', 'Ertiga VDI',
'HECTOR SHARP HYBIRD PETROL MT', 'Dzire VXI AMT',
'TUV300 T 10 MT Dual Tone', 'City ZX CVT',
'Ertiga ZDI PLUS 1.5 DIESEL', 'CRV 2.4 MT',
'Polo TRENDLINE 1.2L PETROL', 'Q3 35 TDI Quattro',
'Creta 1.6 SX PLUS DIESEL', 'Swift Dzire ZDI AMT',
'Ecosport 1.0 ECOBOOST TITANIUM', 'Xcent SX 1.1 CRDI OPT',
'Duster RXL AMT 110 PS', 'Amaze 1.5 VXMT I DTEC',
'Verna FLUIDIC 1.4 EX CRDI', 'Polo COMFORTLINE 1.5L DIESEL',
'Verna VGT CRDI', 'Safari 4X2 EX DICOR BS IV',
'Elite i20 ASTA 1.2 AT', 'Accent EXECUTIVE', 'Indica V2 LS',
'Benz GLA Class 200 CDI SPORT', 'Innova 2.5 E', 'Maxximo 7 Seater',
'Indigo ECS LS CR4 BS IV', 'Etios VX D',
'Safari Storme 2.2 VX 4X4', 'Verna FLUIDIC 1.6 SX CRDI OPT',
'Sail UVA 1.3 Base', 'Alto K10 VXI MUSIK EDITION', 'Micra XV CVT',
'XL6 ZETA SHVS', 'Vento COMFORTLINE PETROL', 'Jazz VX 1.2',
'NEW SANTRO SPORTZ 1.1', 'i10 MAGNA O WITH SUNROOF',
'Ameo COMFORTLINE 1.2', 'Compass LIMITED (O) 2.0',
'Innova Crysta 2.8 GX AT 8 STR', 'CRV 2.4 AT 4WD AVN',
'Ciaz ZETA 1.3 DDIS SHVS', 'Hexa Varicor 400 XTA',
'Ameo HIGHLINE PLUS DSG 1.5', 'Dzire ZXI',
'Benz C Class 250 CDI ELEGANCE AT', 'Duster RXZ 110 4WD',
'Brio 1.2 VX MT I VTEC', 'Vento COMFORTLINE DIESEL',
'Celerio VXI CNG', 'City V MT DIESEL',
'Verna FLUIDIC 1.6 CRDI S AT', 'Benz CLA Class CLA 200 CDI SPORT',
'Innova Crysta 2.8 GX AT 7 STR', 'Benz E Class 200 ELEGANCE',
'Octavia ELEGANCE 1.8 TSI AT', 'Xcent SX 1.2', 'Civic 1.8V AT',
'Vento TRENDLINE PETROL', 'Verna FLUIDIC 1.6 VTVT S (O) MT',
'Amaze 1.2 SX MT I VTEC', 'X1 SDRIVE 20D', 'Manza EX QUADRAJET',
'Fabia AMBIENTE 1.2 MPI', 'Ciaz ZDI SHVS HYBIRD',
'Vento HIGHLINE PETROL', 'A4 2.0 TDI', 'Creta 1.6 SX AT CRDI',
'Beat LT DIESEL', 'Amaze 1.5 SX MT I DTEC',
'Safari Storme 2.2 VX 4X2', 'Ecosport 1.5 TREND TDCI',
'Wagon R 1.0 VXI PLUS', 'Brio 1.2 E MT I VTEC',
'Swift VXI LIMITED EDITION', 'New Wagon-R ZXI 1.2',
'SELTOS HTX 1.5 DIESEL', 'Terrano XL P', 'Swift Dzire VXI OPT',
'IGNIS DELTA 1.2 K12', 'New Elantra SX 1.8 AT', 'Redi Go A',
'New Wagon-R VXI 1.2L', 'Rapid ELEGANCE 1.6 TDI MT',
'Ecosport 1.5 TITANIUM TI VCT', 'Ameo COMFORTLINE 1.0',
'Elite i20 1.2 ASTA (O) CVT', 'WR-V 1.2 i-VTEC S MT',
'Compass 2.0 SPORT', 'Verna 1.6 CRDI SX', 'Go T',
'i20 MAGNA 1.4 CRDI', 'SX4 ZXI MT BS IV',
'3 Series 320 D PERFORMANCE EDITION',
'New Wagon-R VXI (O)1.OL AGS', 'VENUE 1.0L Turbo GDI SX(O) MT',
'Innova Crysta 2.4 GX 7 STR', 'NEXON XM 1.5', '3 Series 320D',
'XUV500 W10 FWD', 'City VX MT O DIESEL', 'Creta 1.6 SX (O) VTVT',
'Ameo HIGHLINE 1.5', 'Ertiga ZXI', 'Creta 1.6 CRDI SX PLUS AUTO',
'Civic 1.8S MT', 'SX4 VXI', 'Alto K10 LXI CNG',
'Polo HIGHLINE1.5L DIESEL', 'Ertiga ZDI SHVS',
'Vitara Brezza ZDI PLUS DUAL TONE',
'Ecosport 1.5 TITANIUMTDCI OPT', 'Jetta COMFORTLINE 2.0L TDI',
'Ciaz ZXI PLUS', 'Rapid 1.5 TDI MT ELEGANCE',
'Elite i20 SPORTZ 1.4', 'Ertiga VXI CNG', 'Kwid 1.0 RXT Opt',
'Bolt XT REVOTRON', 'PUNTO PURE 1.2 Fire',
'i10 ASTA 1.2 AT KAPPA2 WITH SUNROOF', 'Vitara Brezza ZXI',
'Wagon R 1.0 VXI PLUS AMT', 'Kuv100 K8 6 STR',
'Kwid 1.0 CLIMBER OPT AMT', 'Elite i20 Magna Executive Diesel',
'New Figo 1.5 TREND', 'A Star ZXI', 'Grand Punto DYNAMIC 1.3',
'New Figo 1.5 TITANIUM', 'Micra XL CVT (PETROL)',
'Ciaz ALPHA 1.4 VVT AMT', 'Santro Xing GL PLUS LPG',
'i20 ASTA 1.4 AT O WITH SUNROOF', 'Ciaz DELTA 1.4 VVT',
'Jazz 1.2 VX AT', 'Santro Xing XL ERLX EURO III',
'WR-V 1.5 i-DTEC VX MT', 'Figo Aspire 1.5 TITANIUM SPORTS EDITION',
'XUV 3OO W8(O)', 'Ritz ZDI', 'Verna FLUIDIC 1.6 CRDI SX OPT AT',
'Hexa Varicor 400 XT', 'Duster 85 PS RXE DIESEL ADVENTURE',
'Benz GLA Class 200 CGI SPOTRS', 'Baleno SIGMA DDIS 190',
'FREESTYLE TITANIUM 1.5 TDCI', 'Xcent 1.2 S CRDI',
'Grand Punto EMOTION PACK 1.3 90 HP', 'Rapid 1.5 TDI AT AMBITION',
'Polo TRENDLINE 1.5L DIESEL', 'Q3 2.0 TDI',
'Creta 1.6 SX PLUS PETROL', 'Duster RXL PLUS DIESEL 85',
'Innova Crysta 2.4 ZX 7 STR', 'Swift ZDI AMT',
'Xylo H8 ABS AIRBAG BS IV', 'Tiago XZA+ 1.2 RTN',
'Eon MAGNA PLUS OPTIONAL', 'Figo Aspire 1.2 TITANIUM PETROL',
'Jazz 1.2 V MT', 'Omni STD', 'Elite i20 1.2 MAGNA PLUS VTVT',
'City ZX 1.5 EXI', 'Jazz 1.2 SV MT', 'Sunny XV PETROL',
'i20 Magna O 1.4 CRDI', 'Bolero ZLX',
'IGNIS ALPHA 1.2 K12 DUAL TONE', 'Amaze 1.2 V CVT I VTEC',

```
'Grand Punto ACTIVE 1.3', 'City ZX GXI', 'Ciaz ZXI',
'FREESTYLE TREND 1.2 TI-VCT', 'Lodgy 85 PS RXL',
'Rapid Style 1.5 TDI AT', 'Etios CROSS 1.5 V', '5 Series 525D',
'Punto EVO EMOTION 1.3 MULTIJET', 'Jazz 1.5 SV I DTEC',
'Benz A Class A180 CDI STYLE', 'Ertiga VXI ABS',
'Ciaz ALPHA 1.3 DDIS SHVS', 'Wagon R 1.0 VXI OPT',
'Captur RXT Diesel Dual Tone', 'S Cross ZETA 1.3 SHVS',
'Grand i10 Sportz1.2 CRDI', 'Duster RXE PETROL 104',
'XUV 3OO W4 PETROL', 'Tiago XE 1.05 REVOTORQ',
'Scorpio S6+ INTELL HYBRID', 'Creta 1.6 CRDI sx(o) executive',
'Eeco 5 STR', 'VENUE S MT 1.2 KAPPA', 'Swift VDI OPT',
'Dzire ZDI Plus', 'XL6 ALPHA SHVS  MT', 'Corolla HE 1.8 J',
'TIGOR XZ 1.2 REVOTRON', 'AURA S MT', 'Ertiga ZXI Plus SHVS',
'XUV500 W5 FWD', 'S Cross DELTA SHVS',
'Benz GLA Class 200 CDI STYLE', 'Vento TRENDLINE DIESEL',
'Creta 1.6 SX (O)', 'Evalia XE', 'Redi Go S 1.0',
'Ertiga VXI SMART HYBRID', 'Accent EXECUTIVE GLE',
'Ecosport 1.5 ECOSPORT TITANIUM SPORTS(SUNROOF)',
'Amaze 1.2 S (O) MT I VTEC', 'Dzire ZDI', '800 AC',
'Verna SX 1.6 CRDI', 'Fiesta 1.4 EXI DuraTorq',
'A4 35 TDI TECHNOLOGY', 'S60 SUMMUM D4', 'Scala RXZ',
'Swift Dzire TOUR', 'XC 60 SUMMUM D3', 'Ciaz ZDI SHVS',
'Indigo CS LX TDI', 'Fiesta 1.4 ZXI TDCI', 'Aveo U VA LS 1.2',
'Alto 800 LXI UTSAV LIMITED ADDITION', 'Enjoy 1.3 LS 8 STR',
'Xylo D2 BS IV', 'Swift VDI Glory edition',
'Indica V2 DLG DICOR BS III', 'Rapid AMBITION 1.6 TDI MT',
'Fiesta 1.4 EXI', 'Fluence 1.5 E2', 'Spark PS 1.0',
'Indica V2 DLG BS III', 'Fiesta 1.4 SXI TDCI ABS',
'BR-V 1.5 i- DTEC V', 'Creta 1.4 BASE'], dtype=object)
```

In [10]:
```
data.groupby('model_name').count()
```

Out[10]:

model_name	model_year	maker	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
3 Series 320 D PERFORMANCE EDITION	1	1	1	1	1	1	1
3 Series 320D	2	2	2	2	2	2	2
3 Series 320D LUXURYLINE	2	2	2	2	2	2	2
3 Series 320D SPORTLINE	1	1	1	1	1	1	1
5 Series 520D 2.0	3	3	3	3	3	3	3
...	...	...	...	...	...	...	...
i20 Magna O 1.4 CRDI	1	1	1	1	1	1	1
i20 SPORTZ 1.2 O	21	21	21	21	21	21	21
i20 SPORTZ 1.2 VTVT	15	15	15	15	15	15	15
i20 SPORTZ 1.4 CRDI	8	8	8	8	8	8	8
i20 SPORTZ O 1.4 CRDI	5	5	5	5	5	5	5

677 rows × 7 columns

In [11]:
```
data['model_name1'] = data['model_name'].str.extract(r'(\S{3,}|\S{1,2}\s+\S+)', expand=True)
```

In [12]:
```
data
```

Out[12]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
0	2012	Maruti	Alto K10 VXI	Mumbai	29067	Petrol	2nd Owner	165199	Alto
1	2011	Hyundai	i20 SPORTZ 1.2 O	Mumbai	36791	Petrol	2nd Owner	326099	i20
2	2010	Maruti	A Star VXI	Mumbai	35171	Petrol	1st Owner	195199	A Star
3	2011	Hyundai	Santro Xing GLS	Mumbai	19908	Petrol	1st Owner	195199	Santro
4	2012	Hyundai	Santro Xing GLS	Mumbai	43847	Petrol	3rd Owner	203299	Santro
...	...	...	...	...	...	...	...	...	...
3360	2014	Honda	City S MT DIESEL	Kolkata	61643	Diesel	3rd Owner	500000	City
3361	2006	Maruti	Wagon R LXI	Kolkata	26500	Petrol	1st Owner	100000	Wagon
3362	2016	Maruti	S Cross ZETA 1.3	Kolkata	57828	Diesel	1st Owner	550000	S Cross
3363	2012	BMW	3 Series 320D	Kolkata	23782	Diesel	2nd Owner	1200000	3 Series
3364	2016	Hyundai	Creta 1.4 BASE	Kolkata	33130	Diesel	1st Owner	850000	Creta

3365 rows × 9 columns

In [13]:
```
data['model_name1'].unique()
```

Out[13]:
```
array(['Alto', 'i20', 'A Star', 'Santro', 'Eon', 'Wagon', 'i10', 'Swift',
       'Brio', 'Etios', 'Zen', 'Figo', 'Grand', 'Jetta', 'Celerio',
       'Amaze', 'Ritz', 'SELTOS', 'Kuv100', 'Baleno', 'Polo', 'Verna',
       'NEW', 'Go Plus', 'Benz', 'Elite', 'TIGOR', 'City', 'Jazz',
       'Vento', 'Ertiga', 'Kwid', 'Redi', 'IGNIS', 'WR-V', 'TUV300',
       'Vitara', 'Innova', 'Rapid', 'Dzire', 'Creta', 'VENUE', 'XUV500',
       'Sunny', 'Corolla', 'Terrano', 'Fortuner', 'NEXON', 'Tiago',
       'Compass', 'Ciaz', 'Scorpio', 'Duster', 'Superb', 'Accord',
       'HECTOR', 'Ecosport', 'S Cross', 'Yeti', 'New', '5 Series',
       'Prius', 'Eeco', 'Ameo', 'Quanto', 'Xcent', 'Manza', 'Civic',
       'Hexa', '3 Series', 'Santa', 'Mobilio', 'Beat', 'Laura', 'Rexton',
       'Punto', 'Cruze', 'Camry', 'CRV', 'S PRESSO', 'Nano', 'TRIBER',
       'Fiesta', 'YARIS', 'GRAND', 'Spark', 'OMNI', 'Micra', 'E2O',
       'Indica', 'A3 35TDI', 'Omni', 'Pulse', 'Thar', 'Indigo', 'BR-V',
       'Q3 35', 'Safari', 'Accent', 'Maxximo', 'Sail', 'XL6', 'Octavia',
       'X1 SDRIVE', 'Fabia', 'A4 2.0', 'Go T', 'SX4', 'Bolt', 'PUNTO',
       'XUV', 'FREESTYLE', 'Q3 2.0', 'Xylo', 'Bolero', 'Lodgy', 'Captur',
       'AURA', 'Evalia', '800', 'A4 35', 'S60', 'Scala', 'XC 60', 'Aveo',
       'Enjoy', 'Fluence'], dtype=object)
```

In [14]:
```
data['model_name1'].value_counts()
```

Out[14]:
```
model_name1
Swift      465
Alto       393
Wagon      234
i10        209
Grand      133
          ...
Sail         1
Octavia      1
Fabia        1
A4 2.0       1
Fluence      1
Name: count, Length: 127, dtype: int64
```

In [15]:
```
dropping=data.drop(['city','model_name'],axis=1)
```

In [16]:
```
dropping
```

Out[16]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
0	2012	Maruti	29067	Petrol	2nd Owner	165199	Alto
1	2011	Hyundai	36791	Petrol	2nd Owner	326099	i20
2	2010	Maruti	35171	Petrol	1st Owner	195199	A Star
3	2011	Hyundai	19908	Petrol	1st Owner	195199	Santro
4	2012	Hyundai	43847	Petrol	3rd Owner	203299	Santro
...	...	...	...	...	...	...	...
3360	2014	Honda	61643	Diesel	3rd Owner	500000	City
3361	2006	Maruti	26500	Petrol	1st Owner	100000	Wagon
3362	2016	Maruti	57828	Diesel	1st Owner	550000	S Cross
3363	2012	BMW	23782	Diesel	2nd Owner	1200000	3 Series
3364	2016	Hyundai	33130	Diesel	1st Owner	850000	Creta

3365 rows × 7 columns

In [17]:
```
wagon_car=dropping.loc[(dropping.model_name1=="Wagon")]
```

In [18]:
```
wagon_car
```

Out[18]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
8	2013	Maruti	13688	Petrol	1st Owner	326199	Wagon
14	2011	Maruti	18514	Petrol	1st Owner	269399	Wagon
16	2012	Maruti	20712	Petrol	2nd Owner	258399	Wagon
19	2012	Maruti	39652	Petrol	3rd Owner	288299	Wagon
21	2014	Maruti	6858	Petrol	1st Owner	358399	Wagon
...	...	...	...	...	...	...	...
3314	2014	Maruti	95432	Petrol + CNG	1st Owner	270000	Wagon
3327	2013	Maruti	22008	Petrol	1st Owner	332599	Wagon
3331	2014	Maruti	25852	Petrol	1st Owner	345499	Wagon
3357	2001	Maruti	72000	Petrol	2nd Owner	38000	Wagon
3361	2006	Maruti	26500	Petrol	1st Owner	100000	Wagon

234 rows × 7 columns

In [19]:
```
swift_car=dropping.loc[(dropping.model_name1=='Swift')]
```

In [20]:
```
swift_car
```

Out[20]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
10	2009	Maruti	42533	Petrol	2nd Owner	274899	Swift
12	2015	Maruti	23070	Petrol	2nd Owner	478799	Swift
28	2012	Maruti	50581	Petrol	1st Owner	406299	Swift
36	2012	Maruti	47260	Petrol	1st Owner	364799	Swift
37	2012	Maruti	50525	Diesel	1st Owner	376799	Swift
...	...	...	...	...	...	...	...
3299	2015	Maruti	61537	Diesel	1st Owner	450000	Swift
3322	2017	Maruti	18140	Petrol	1st Owner	457699	Swift
3345	2013	Maruti	78261	Petrol	1st Owner	327699	Swift
3348	2007	Maruti	90265	Diesel	3rd Owner	130000	Swift
3354	2017	Maruti	43860	Petrol	1st Owner	429999	Swift

465 rows × 7 columns

In [21]:
```
i10_car=dropping.loc[(dropping.model_name1=="i10")]
```

In [22]:
```
i10_car
```

Out[22]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
9	2014	Hyundai	13068	Petrol	1st Owner	369699	i10
22	2011	Hyundai	43411	Petrol	2nd Owner	242999	i10
30	2014	Hyundai	24244	Petrol	2nd Owner	273299	i10
35	2011	Hyundai	39413	Petrol	1st Owner	255199	i10
43	2010	Hyundai	28925	Petrol	2nd Owner	226799	i10
...	...	...	...	...	...	...	...
3195	2011	Hyundai	51086	Petrol	1st Owner	259299	i10
3233	2013	Hyundai	178433	Petrol + CNG	1st Owner	267599	i10
3249	2013	Hyundai	47894	Petrol + CNG	1st Owner	265099	i10
3316	2013	Hyundai	55485	Petrol	1st Owner	300000	i10
3346	2016	Hyundai	16014	Petrol	1st Owner	300000	i10

209 rows × 7 columns

In [23]:
```
wagon_car.head(10)
```

Out[23]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
8	2013	Maruti	13688	Petrol	1st Owner	326199	Wagon
14	2011	Maruti	18514	Petrol	1st Owner	269399	Wagon
16	2012	Maruti	20712	Petrol	2nd Owner	258399	Wagon
19	2012	Maruti	39652	Petrol	3rd Owner	288299	Wagon
21	2014	Maruti	6858	Petrol	1st Owner	358399	Wagon
23	2013	Maruti	17781	Petrol	1st Owner	302399	Wagon
24	2012	Maruti	41964	Petrol + CNG	2nd Owner	254699	Wagon
25	2012	Maruti	25748	Petrol	2nd Owner	270599	Wagon
39	2012	Maruti	48921	Petrol	1st Owner	269699	Wagon
40	2013	Maruti	24065	Petrol	1st Owner	312999	Wagon

In [24]:
```
swift_car.head(10)
```

Out[24]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
10	2009	Maruti	42533	Petrol	2nd Owner	274899	Swift
12	2015	Maruti	23070	Petrol	2nd Owner	478799	Swift
28	2012	Maruti	50581	Petrol	1st Owner	406299	Swift
36	2012	Maruti	47260	Petrol	1st Owner	364799	Swift
37	2012	Maruti	50525	Diesel	1st Owner	376799	Swift
71	2012	Maruti	63871	Petrol	1st Owner	340599	Swift
78	2012	Maruti	64239	Petrol	3rd Owner	341699	Swift
79	2014	Maruti	19856	Petrol	1st Owner	467499	Swift
84	2013	Maruti	53065	Diesel	1st Owner	390499	Swift
102	2012	Maruti	45627	Petrol	1st Owner	358099	Swift

In [25]:
```
i10_car.head(10)
```

Out[25]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
9	2014	Hyundai	13068	Petrol	1st Owner	369699	i10
22	2011	Hyundai	43411	Petrol	2nd Owner	242999	i10
30	2014	Hyundai	24244	Petrol	2nd Owner	273299	i10
35	2011	Hyundai	39413	Petrol	1st Owner	255199	i10
43	2010	Hyundai	28925	Petrol	2nd Owner	226799	i10
44	2012	Hyundai	50107	Petrol	3rd Owner	243699	i10
49	2012	Hyundai	48978	Petrol	3rd Owner	275899	i10
57	2012	Hyundai	39357	Petrol	1st Owner	282099	i10
58	2010	Hyundai	41923	Petrol	2nd Owner	227299	i10
103	2010	Hyundai	66817	Petrol	1st Owner	223999	i10

In [26]:
```
con=pd.concat([wagon_car,swift_car,i10_car])
```

In [27]:
```
con
```

Out[27]:

	model_year	maker	distance_covered (km)	fuel_type	pre_owner	price (₹)	model_name1
8	2013	Maruti	13688	Petrol	1st Owner	326199	Wagon
14	2011	Maruti	18514	Petrol	1st Owner	269399	Wagon
16	2012	Maruti	20712	Petrol	2nd Owner	258399	Wagon
19	2012	Maruti	39652	Petrol	3rd Owner	288299	Wagon
21	2014	Maruti	6858	Petrol	1st Owner	358399	Wagon
...	...	...	...	...	...	...	...
3195	2011	Hyundai	51086	Petrol	1st Owner	259299	i10
3233	2013	Hyundai	178433	Petrol + CNG	1st Owner	267599	i10
3249	2013	Hyundai	47894	Petrol + CNG	1st Owner	265099	i10
3316	2013	Hyundai	55485	Petrol	1st Owner	300000	i10
3346	2016	Hyundai	16014	Petrol	1st Owner	300000	i10

908 rows × 7 columns

In [28]:
```
dum=pd.get_dummies(con,dtype=int)
```

In [29]:
```
dum
```

Out[29]:

	model_year	distance_covered (km)	price (₹)	maker_Hyundai	maker_Maruti	fuel_type_Diesel	fuel_type_Petrol	fuel_type_Petrol + CNG	fuel_type_Petrol + LPG	pre_owner_1st Owner	pre_owner_2nd Owner	pre_owner_3rd Owner	pre_owner_4th Owner	model_na
8	2013	13688	326199	0	1	0	1	0	0	1	0	0	0	
14	2011	18514	269399	0	1	0	1	0	0	1	0	0	0	
16	2012	20712	258399	0	1	0	1	0	0	0	1	0	0	
19	2012	39652	288299	0	1	0	1	0	0	0	0	1	0	
21	2014	6858	358399	0	1	0	1	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3195	2011	51086	259299	1	0	0	1	0	0	1	0	0	0	
3233	2013	178433	267599	1	0	0	0	1	0	1	0	0	0	
3249	2013	47894	265099	1	0	0	0	1	0	1	0	0	0	
3316	2013	55485	300000	1	0	0	1	0	0	1	0	0	0	
3346	2016	16014	300000	1	0	0	1	0	0	1	0	0	0	

908 rows × 16 columns

In [30]:
```
y=dum['price (₹)']
y
```

Out[30]:
```
8       326199
14      269399
16      258399
19      288299
21      358399
         ...
3195    259299
3233    267599
3249    265099
3316    300000
3346    300000
Name: price (₹), Length: 908, dtype: int64
```

In [31]:
```
x=dum.drop(['price (₹)'],axis=1)
```

In [32]:
```
x
```

Out[32]:

	model_year	distance_covered (km)	maker_Hyundai	maker_Maruti	fuel_type_Diesel	fuel_type_Petrol	fuel_type_Petrol + CNG	fuel_type_Petrol + LPG	pre_owner_1st Owner	pre_owner_2nd Owner	pre_owner_3rd Owner	pre_owner_4th Owner	model_name1_Swi
8	2013	13688	0	1	0	1	0	0	1	0	0	0	
14	2011	18514	0	1	0	1	0	0	1	0	0	0	
16	2012	20712	0	1	0	1	0	0	0	1	0	0	
19	2012	39652	0	1	0	1	0	0	0	0	1	0	
21	2014	6858	0	1	0	1	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3195	2011	51086	1	0	0	1	0	0	1	0	0	0	
3233	2013	178433	1	0	0	0	1	0	1	0	0	0	
3249	2013	47894	1	0	0	0	1	0	1	0	0	0	
3316	2013	55485	1	0	0	1	0	0	1	0	0	0	
3346	2016	16014	1	0	0	1	0	0	1	0	0	0	

908 rows × 15 columns

In [33]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.34,random_state=42)
```

In [34]:
```
#LINEAR REGRESSION
```

In [35]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
```

In [36]:
```
reg.fit(x_train,y_train)
```

Out[36]:
```
LinearRegression
LinearRegression()
```

In [37]:
```
ypred=reg.predict(x_test)
```

In [38]:
```
ypred
```

Out[38]:
```
array([268681.50831824, 431006.63104398, 376909.35676789, 356349.71561378,
       189455.73403636, 268125.88651438, 333117.71419929, 328209.53120726,
       295084.41139907, 203670.3315887 , 545013.06502353, 346176.43335059,
       347921.58741903, 261966.18634459, 392801.97738311, 281964.22789361,
       222924.98808407, 409235.46730509, 446478.54345123, 371543.22919726,
       446098.2624102 , 177331.62458451, 410773.71721964, 403089.53374575,
       473151.59045965, 246554.66036487, 452660.23815782, 503873.78756075,
       403260.81979056, 387018.75800737, 359183.84255699, 419115.49775692,
       415916.57896916, 264849.74791986, 512216.64857572, 308932.30789828,
       457271.28659619, 259466.940535  , 457355.64622558, 546489.35853786,
       313576.65314224, 420520.45304822, 258206.43650744, 311721.96389896,
       294324.56343289, 465749.5227029 , 377767.92995252, 322953.40933952,
       395783.9582987 , 421217.15287174, 291955.76949166, 500494.36984619,
       383107.34432145, 444078.57312524, 434053.13251137, 314017.09598989,
       288956.61135937, 198865.69399694, 205927.86862739, 307675.77733152,
       269552.66459408, 345906.73714025, 300887.19242799, 546134.80357376,
       458772.43118078, 300515.71449089, 344065.78169734, 307681.72682295,
       259539.98168155, 429418.98727417, 471534.39191217, 323198.65348336,
       446216.94565119, 317014.42021719, 329787.64839417, 397122.09872866,
       245312.24114347, 380531.0134659 , 400720.47578885, 244447.69700863,
       442830.2437238 , 181911.19098727, 331357.04383262, 215701.81787334,
       285356.35021954, 380112.57747808, 343631.14389171, 407209.30794556,
       295610.93560381, 422648.82136473, 348356.2229009 , 305746.57255816,
       404272.30910928, 414611.25412294, 547501.67409055, 405670.66166168,
       172712.61982244, 205063.37589968, 380464.59651703, 380815.78932199,
       231062.33016114, 290328.25071008, 381249.46590555, 250539.78730821,
       376369.27174929, 441175.14447325, 259624.76854115, 320386.39818459,
       329448.34743452, 345339.03001753, 422570.29191194, 472831.9704088 ,
       428522.53619728, 458943.7172256 , 371891.33531824, 286761.53926751,
       402451.54710826, 395308.36560547, 240452.21994206, 307597.6956552 ,
       356014.41665204, 226511.06770629, 454392.97271948, 435525.2795784 ,
       497459.04216079, 375778.66362504, 486691.57468189, 238973.05230802,
       368437.9611432 , 259628.81389612, 364801.32755446, 410382.26922348,
       552111.19427476, 447012.6370326 , 515549.15958749, 335237.87330043,
       427435.619031  , 484333.99910291, 399583.31970897, 433245.1243782 ,
       337867.62115727, 415089.95473161, 326581.98247732, 323550.39681983,
       470171.80050701, 444421.15629981, 322555.8697855 , 376822.1555625 ,
       256408.21114435, 414912.3201916 , 365783.88144722, 297142.24626775,
       203717.27855445, 466691.9047581 , 456470.34875432, 435347.48414762,
       375961.97417948, 218027.45027356, 307750.03134848, 432386.8568444 ,
       433117.69079243, 354751.40664766, 378814.17274746, 438338.48982809,
       381440.84527165, 331719.66729772, 259152.41642768, 429665.34184406,
       324604.84078009, 218174.3898802 , 573264.11710078, 467090.2288276 ,
       403125.81284172, 526510.01332398, 390343.70196629, 283452.74744119,
       349058.10938644, 442714.79842815, 378481.45398045, 350741.07194598,
       270397.12643335, 219705.75195127, 337111.32882378, 266522.27866989,
       462635.36796393, 455151.9970861 , 322111.45347705, 298713.40371703,
       336516.72442792, 282691.25904357, 248323.81878255, 427858.12183255,
       286268.65691391, 339987.50314467, 333897.73512032, 390542.82137174,
       326659.80185988, 304646.77759507, 305618.60652366, 286227.83128121,
       367967.5645141 , 451819.57679284, 451382.61768372, 303328.5986027 ,
       307237.16838886, 314078.21675401, 311019.88393987, 365379.76910266,
       494529.59387677, 388653.89923538, 203010.55302368, 501890.41931758,
       375806.26291835, 231294.43131924, 285486.24242067, 172280.55112661,
       303735.29827955, 402616.69191001, 437211.34291583, 421027.47636993,
       372927.33773997, 450788.95305116, 196670.10241323, 450462.21231268,
       289134.3785637 , 263888.29198452, 358979.36215097, 404187.94947989,
       253053.00001375, 451366.74665131, 278841.40275409, 268822.59742284,
       474920.53394502, 383456.70322143, 227270.06771898, 324887.67997617,
       408285.02820838, 273121.697942  , 285840.73357973, 290673.63614925,
       298631.56923851, 345029.63143221, 458909.57505102, 206559.3432445 ,
       363710.03250796, 442670.38833912, 280468.92424195, 401269.81027661,
       422131.04885681, 355560.78918349, 303720.13731737, 472056.1378809 ,
       474875.4611472 , 459634.48810394, 397522.50022614, 454498.42225622,
       371993.62585801, 342129.17570756, 432699.32534675, 232617.78724074,
       306392.33035662, 187175.45557392, 410404.02183962, 395275.78254592,
       361882.52337786, 358599.22214991, 188561.09405365, 253830.92328049,
       366412.08462369, 287742.72751877, 257752.00426369, 295031.70446802,
       403897.06324159, 366951.12002548, 376019.44476458, 465599.26407336,
       449871.78562547, 577997.54745098, 365883.63714748, 255545.29707278,
       370084.21565466, 381070.58053577, 220289.96286684, 235442.00092034,
       273205.31360541, 276386.65989278, 360503.89600384, 208156.49041651,
       312781.96098131, 292470.98461565, 272692.56417511, 268578.50398815,
       232872.39438307, 437578.29798535, 273726.3360952 , 284081.16554998,
       396397.54390695, 436705.09178655, 502784.61152381, 168558.40719932,
       489177.34217292, 391492.45936904, 145922.65061212, 267649.64196999,
       408901.30022816, 393700.15251146, 305356.05245972, 514394.01437815,
       410498.89517665])
```

In [39]:
```
from sklearn.metrics import r2_score
```

In [40]:
```
r2_score(y_test,ypred)#0.30= 0.78,0.35=0.74,0.25=0.79
```

Out[40]:
```
0.7434714573446568
```

In [41]:
```
from sklearn.metrics import mean_squared_error#average difference error between estimated value and actual value
mean_squared_error(ypred,y_test)
```

Out[41]:
```
2664423718.427918
```

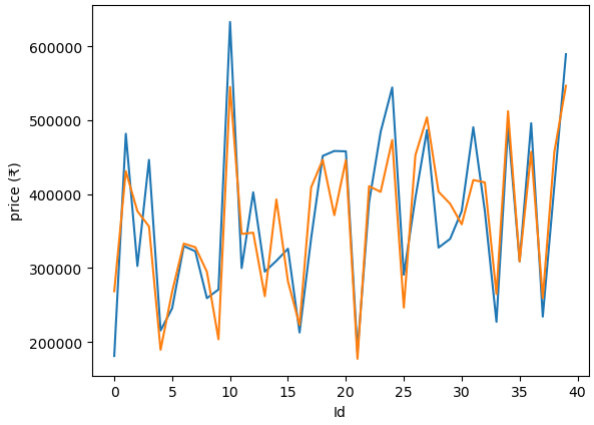
In [42]:
```
Results = pd.DataFrame(columns=['price (₹)','predicted'])
Results['price (₹)']=y_test
Results['predicted']=ypred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[42]:

	index	price (₹)	predicted	Id
0	2460	181099	268681.508318	0
1	1123	481699	431006.631044	1
2	662	302799	376909.356768	2
3	1017	446399	356349.715614	3
4	1414	215499	189455.734036	4
5	1900	245499	268125.886514	5
6	1761	329699	333117.714199	6
7	297	322799	328209.531207	7
8	3195	259299	295084.411399	8
9	1013	270999	203670.331589	9
10	1845	632899	545013.065024	10
11	3316	300000	346176.433351	11
12	1165	402499	347921.587419	12
13	2062	295199	261966.186345	13
14	559	309799	392801.977383	14

In [43]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='price (₹)',data=Results.head(40))
sns.lineplot(x='Id',y='predicted',data=Results.head(40))
plt.plot()
```

Out[43]:
```
[]
```



In [44]:
```
#RIDGE
```

In [45]:
```
from sklearn.model_selection import GridSearchCV#is used to find the optimal parameter values
#from sklearn.grid_search import GridSearchCV


from sklearn.linear_model import Ridge#its correct the overfitting of the training data

alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]


ridge = Ridge()

parameters = {'alpha': alpha}

ridge_regressor = GridSearchCV(ridge, parameters)

ridge_regressor.fit(x_train, y_train)
```

```
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=9.95945e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.34093e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=8.71085e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.28422e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.43332e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=9.98101e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.34231e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=8.7099e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.28514e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=1.43536e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[45]:
```
GridSearchCV
estimator: Ridge
Ridge
```

In [46]:
```
ridge_regressor.best_params_#to find the best parameters
```

Out[46]:
```
{'alpha': 5}
```

In [47]:
```
ridge=Ridge(alpha=5)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [48]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[48]:
```
0.7437428912090313
```

In [49]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[49]:
```
2661604481.165906
```

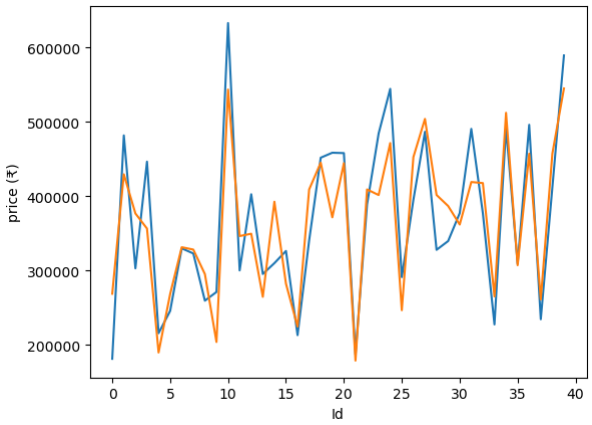
In [50]:
```
Results= pd.DataFrame(columns=['price (₹)','Predicted'])
Results['price (₹)']=y_test
Results['Predicted']=y_pred_ridge
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[50]:

	index	price (₹)	Predicted	Id
0	2460	181099	268634.342230	0
1	1123	481699	429392.801653	1
2	662	302799	376667.190827	2
3	1017	446399	356359.669772	3
4	1414	215499	189573.201621	4
5	1900	245499	268349.226465	5
6	1761	329699	331234.645475	6
7	297	322799	328053.853481	7
8	3195	259299	295071.563453	8
9	1013	270999	203710.783444	9

In [51]:
```
sns.lineplot(x='Id',y='price (₹)',data=Results.head(40))
sns.lineplot(x='Id',y='Predicted',data=Results.head(40))
plt.plot()
```

Out[51]:
```
[]
```

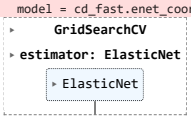


In [52]:
```
#ELASTICNET
```

In [53]:
```
from sklearn.linear_model import ElasticNet#combines both lasso and ridge regression to improve the statistical model

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(x_train, y_train)
```

```
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.106e+11, tolerance: 5.212e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.848e+11, tolerance: 4.995e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.111e+11, tolerance: 5.043e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.782e+11, tolerance: 4.901e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.633e+11, tolerance: 5.114e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.105e+11, tolerance: 5.212e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.848e+11, tolerance: 4.995e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.111e+11, tolerance: 5.043e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.782e+11, tolerance: 4.901e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.633e+11, tolerance: 5.114e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.105e+11, tolerance: 5.212e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.848e+11, tolerance: 4.995e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.111e+11, tolerance: 5.043e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.782e+11, tolerance: 4.901e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.633e+11, tolerance: 5.114e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.106e+11, tolerance: 5.212e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.849e+11, tolerance: 4.995e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.112e+11, tolerance: 5.043e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.783e+11, tolerance: 4.901e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.634e+11, tolerance: 5.114e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.112e+11, tolerance: 5.212e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.857e+11, tolerance: 4.995e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 6.119e+11, tolerance: 5.043e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.790e+11, tolerance: 4.901e+08
  model = cd_fast.enet_coordinate_descent(
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 5.641e+11, tolerance: 5.114e+08
  model = cd_fast.enet_coordinate_descent(
```

Out[53]:



In [54]:
```
elastic_regressor.best_params_
```

Out[54]:
```
{'alpha': 0.01}
```

In [55]:
```
elastic=ElasticNet(alpha=0.001)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

```
C:\Users\pedda\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 7.415e+11, tolerance: 6.318e+08
  model = cd_fast.enet_coordinate_descent(
```

In [56]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[56]:
```
0.7434875232107913
```

In [57]:
```
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[57]:
```
2664256850.9349585
```

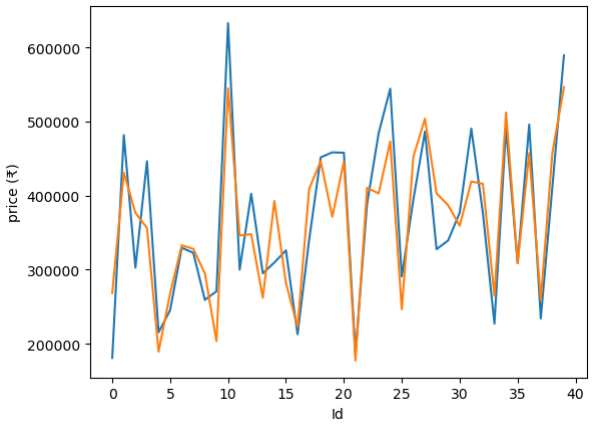
In [58]:
```
Results= pd.DataFrame(columns=['price (₹)','Predicted'])
Results['price (₹)']=y_test
Results['Predicted']=y_pred_elastic
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[58]:

	index	price (₹)	Predicted	Id
0	2460	181099	268681.398414	0
1	1123	481699	430902.856029	1
2	662	302799	376899.668602	2
3	1017	446399	356350.738241	3
4	1414	215499	189469.319929	4
5	1900	245499	268143.146385	5
6	1761	329699	332998.480589	6
7	297	322799	328204.966717	7
8	3195	259299	295085.475450	8
9	1013	270999	203679.902654	9

In [59]:
```
sns.lineplot(x='Id',y='price (₹)',data=Results.head(40))
sns.lineplot(x='Id',y='Predicted',data=Results.head(40))
plt.plot()
```

Out[59]:
```
[]
```



In [60]:
```
#RANDOMFORESTREGRESSOR
```

In [61]:
```
from sklearn.model_selection import GridSearchCV #GridSearchCV is for parameter tuning
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
n_estimators=[25,50,75,100,125,150,175,200] #number of decision trees in the forest, default = 100
criterion=['squared_error'] #criteria for choosing nodes default = 'gini'
max_depth=[3,5,10] #maximum number of nodes in a tree default = None (it will go till all possible nodes)
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth}
RFC_reg = GridSearchCV(reg, parameters)
RFC_reg.fit(x_train,y_train)
```

Out[61]:

GridSearchCV
estimator: RandomForestRegressor
RandomForestRegressor

In [62]:
```
RFC_reg.best_params_
```

Out[62]:
```
{'criterion': 'squared_error', 'max_depth': 5, 'n_estimators': 150}
```

In [63]:
```
reg=RandomForestRegressor(n_estimators=50,criterion='squared_error',max_depth=5)
```

In [64]:
```
reg.fit(x_train,y_train)
```

Out[64]:

RandomForestRegressor
RandomForestRegressor(max_depth=5, n_estimators=50)

In [65]:
```
ypred=reg.predict(x_test)
ypred
```

Out[65]:
```
array([269927.82267   , 438892.367093  , 360545.59924264, 331061.0189189 ,
       229431.63064974, 287402.60532732, 272063.38072582, 351364.79059325,
       288652.79330431, 234443.38338796, 547586.67720582, 304588.80049199,
       343380.4736257 , 290923.59005179, 406393.66903793, 291761.5426158 ,
       243830.15240123, 403747.01003318, 450370.2094745 , 362939.23808766,
       459076.56646568, 199611.91332411, 403571.15104329, 404393.78824191,
       474871.35705706, 247842.69039053, 445355.01952991, 561312.87511361,
       379399.13746293, 390036.5363854 , 342783.03948137, 435424.43014179,
       385388.93033315, 271140.32654693, 561995.47511361, 253769.21183198,
       448210.0025991 , 286792.14740035, 448210.0025991 , 550118.26591711,
       302702.90556286, 430369.61861878, 287324.96646868, 304091.8975143 ,
       289312.04330431, 460359.36025259, 359631.13737368, 328787.72205099,
       385018.27600518, 441042.796986  , 291014.46205226, 507703.67747796,
       389408.61822252, 450131.5878872 , 446644.3016345 , 305301.81942155,
       300510.25731865, 209795.74689004, 234047.2950307 , 250134.18199793,
       289173.80837206, 340171.31926803, 295290.4922401 , 547946.76591711,
       451721.99983153, 314172.61484428, 310807.23817571, 273152.97054428,
       266582.35985601, 435540.68027705, 463437.06927957, 300812.56376837,
       450836.40298023, 297319.31362076, 270074.30090503, 405478.19043722,
       266508.70619706, 373066.38189112, 342425.77175588, 251313.67236965,
       457270.37631561, 201054.16175256, 271761.92994218, 242366.20795403,
       294938.86910829, 373066.38189112, 354809.97489899, 395951.16778064,
       304891.87574592, 434675.66385935, 344432.54852766, 297239.35422368,
       379888.86400139, 411073.20723156, 548838.53258378, 382943.32175794,
       214694.34882052, 210759.43104504, 383468.92249879, 383468.92249879,
       243830.15240123, 301786.36856734, 368285.01644281, 266508.70619706,
       360545.59924264, 461834.20964894, 267725.83919026, 352072.19059325,
       298468.87830056, 345552.05212452, 394844.10717711, 455997.70700221,
       415373.82838923, 454612.6999414 , 373148.99429908, 290981.66815287,
       380803.32587035, 385018.27600518, 253918.7422487 , 304226.44582471,
       331061.0189189 , 242991.53333291, 446145.75286324, 439324.98164317,
       496875.53224739, 360055.87270417, 479222.57465272, 275547.44971964,
       338403.95195638, 291656.35905696, 333538.39484051, 403747.01003318,
       567222.37261004, 454620.62410572, 539183.65867208, 328763.86687547,
       434900.3827829 , 471869.01545315, 380803.32587035, 437810.65165752,
       301070.91543942, 431109.96318729, 271018.00595554, 316784.2353416 ,
       464678.68692663, 457173.64660505, 304181.34340497, 359141.41083522,
       275940.57872157, 413079.73100056, 358397.0754741 , 304891.87574592,
       241813.95735106, 476958.78893843, 487330.59685873, 453523.3105222 ,
       338413.61323334, 241813.95735106, 303318.31512624, 437810.65165752,
       433902.32626367, 327505.53811805, 359631.13737368, 438362.67861287,
       386320.82114258, 272063.38072582, 266582.35985601, 435540.68027705,
       338124.54898786, 214149.82984113, 616271.1457458 , 468097.63232345,
       380803.32587035, 536972.56425128, 367592.5270358 , 291761.5426158 ,
       316534.28547018, 455496.96602412, 341839.09379257, 370342.2676426 ,
       270800.69767   , 243830.15240123, 327418.01742843, 291233.87429644,
       454612.6999414 , 450591.63728444, 302472.18017264, 292316.4339575 ,
       345944.94915499, 301484.81051526, 266508.70619706, 438251.97424288,
       293788.52158304, 328627.93933568, 272063.38072582, 406393.66903793,
       266387.28149886, 248556.82088682, 302836.34149158, 250323.13178861,
       338403.95195638, 444188.93994214, 445907.13127594, 294374.72560307,
       250134.18199793, 296484.67765681, 304091.8975143 , 362882.81204384,
       491729.90161924, 403410.95148798, 237046.37433931, 561312.87511361,
       360545.59924264, 238893.94351494, 291436.94681278, 210469.024797  ,
       295971.34175223, 405305.50945403, 452985.03179204, 416001.9181324 ,
       360983.6514065 , 452309.82861823, 232724.67316043, 452309.82861823,
       301786.36856734, 269276.01696804, 343375.47892582, 379888.86400139,
       241217.79337703, 409413.44010092, 294373.25614058, 284163.6895404 ,
       477963.27639257, 389408.61822252, 239489.54428467, 302827.8546945 ,
       351654.26601706, 277979.00193376, 339848.61179467, 302599.66182821,
       254484.14505709, 305977.79244344, 448210.0025991 , 233103.5899802 ,
       338751.28717919, 457270.37631561, 291547.28112058, 380313.59933188,
       434675.66385935, 369167.60529725, 294374.72560307, 463437.06927957,
       481774.1444712 , 454612.6999414 , 384103.81413622, 446145.75286324,
       374172.43006971, 355750.72872301, 439382.33678997, 243297.33333291,
       300743.35784592, 240325.01459244, 418473.70028058, 381649.57757561,
       366818.89615534, 360554.55825745, 229431.63064974, 286491.42416368,
       358886.80201256, 250323.13178861, 246412.90321719, 256079.27863462,
       405129.65046413, 365802.12301777, 379171.87039876, 455743.06248849,
       445907.13127594, 617647.28860295, 332872.61088626, 247010.91988385,
       379521.57590278, 398400.69699303, 235071.40809522, 238396.53333291,
       280699.29109721, 282237.02106648, 363922.46722532, 235031.39752113,
       302198.33020676, 300865.93184051, 277979.00193376, 269927.82267   ,
       243297.33333291, 417492.38512257, 291226.56468034, 291969.76588111,
       368065.82257978, 439324.98164317, 525916.38655971, 199364.62916826,
       476252.67555649, 349432.68203037, 183209.40542631, 296051.84623399,
       403747.01003318, 378431.98097221, 302836.34149158, 513701.36173621,
       431272.3176193 ])
```

In [66]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[66]:
```
0.7379071644869787
```

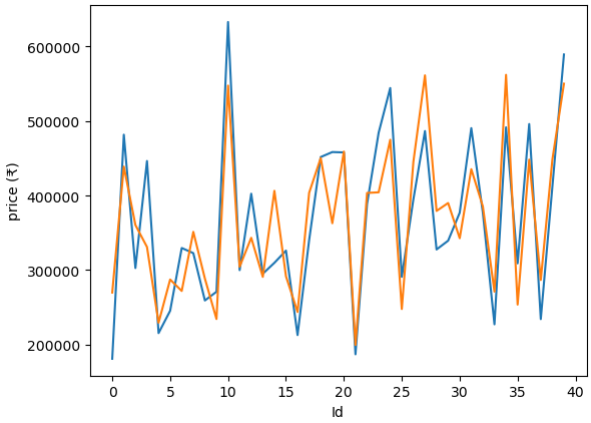
In [67]:
```
Results= pd.DataFrame(columns=['price (₹)','Predicted'])
Results['price (₹)']=y_test
Results['Predicted']=ypred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[67]:

	index	price (₹)	Predicted	Id
0	2460	181099	269927.822670	0
1	1123	481699	438892.367093	1
2	662	302799	360545.599243	2
3	1017	446399	331061.018919	3
4	1414	215499	229431.630650	4
5	1900	245499	287402.605327	5
6	1761	329699	272063.380726	6
7	297	322799	351364.790593	7
8	3195	259299	288652.793304	8
9	1013	270999	234443.383388	9

In [68]:
```
sns.lineplot(x='Id',y='price (₹)',data=Results.head(40))
sns.lineplot(x='Id',y='Predicted',data=Results.head(40))
plt.plot()
```

Out[68]:
```
[]
```


600000
500000
400000
300000
200000
price (₹)
0
5
10
15
20
25
30
35
40
Id

In [ ]:

In [ ]: