In [1]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\carIndia-2021.csv")
a
```

Out[2]:

	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 [3]:
```
a.describe()
```

Out[3]:

	model_year	distance_covered (km)	price (₹)
count	3365.000000	3365.000000	3.365000e+03
mean	2013.876374	60937.813967	4.336549e+05
std	3.035588	41342.775191	2.595909e+05
min	2001.000000	60.000000	2.700000e+04
25%	2012.000000	30598.000000	2.769990e+05
50%	2014.000000	53488.000000	3.647990e+05
75%	2016.000000	82414.000000	4.975990e+05
max	2021.000000	428123.000000	3.600000e+06

In [4]:
```
a.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 [5]:
```
a.head(5)
```

Out[5]:

	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

In [6]:
```
a.tail(5)
```

Out[6]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
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 [7]:
```
list(a)
```

Out[7]:
```
['model_year',
 'maker',
 'model_name',
 'city',
```

```
'distance_covered (km)',
'fuel_type',
'pre_owner',
'price (₹)']
```

In [8]:
```
a.isna().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]:
```
a['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)1.0L 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]:
```
#a[a.duplicated()].count()
```

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

In [12]:
```
a.head(10)
```

Out[12]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)	ref_
0	2012	Maruti	Alto K10 VXI	Mumbai	29067	Petrol	2nd Owner	165199	

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)	ref_
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 [13]:
```
a['ref_model'].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]:
```
#a.groupby(['ref_model']).count().sort_values(by=['ref_model'],ascending=True)
a['ref_model'].value_counts()
```

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

In [15]:
```
#sorting techniques
#a.sort(reverse=True)
#data=data.groupby('ref_model').size().sort_values(ascending=True)#false-decending
#data.head(10)
#data.groupby(['ref_model']).count()
#grouped = data.groupby('ref_model').count().reset_index()
#grouped.sort_values('ref_model', ascending=False)#descending[reverse=True]
```

In [16]:
```
c=a.loc[(a.ref_model == "Wagon")]
c
```

Out[16]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
8	2013	Maruti	Wagon R 1.0 LXI	Mumbai	13688	Petrol	1st Owner	326199
14	2011	Maruti	Wagon R 1.0 VXI	Mumbai	18514	Petrol	1st Owner	269399
16	2012	Maruti	Wagon R 1.0 LXI	Mumbai	20712	Petrol	2nd Owner	258399
19	2012	Maruti	Wagon R 1.0 VXI	Mumbai	39652	Petrol	3rd Owner	288299
21	2014	Maruti	Wagon R 1.0 VXI	Mumbai	6858	Petrol	1st Owner	358399
...	...	...	...	...	...	...	...	...
3314	2014	Maruti	Wagon R 1.0 VXI	Ahmedabad	95432	Petrol + CNG	1st Owner	270000
3327	2013	Maruti	Wagon R Stingray VXI	Kolkata	22008	Petrol	1st Owner	332599
3331	2014	Maruti	Wagon R Stingray VXI	Kolkata	25852	Petrol	1st Owner	345499
3357	2001	Maruti	Wagon R LXI	Kolkata	72000	Petrol	2nd Owner	38000
3361	2006	Maruti	Wagon R LXI	Kolkata	26500	Petrol	1st Owner	100000

234 rows × 9 columns

In [17]:
```
d=a.loc[(a.ref_model=='Grand')]
d
```

Out[17]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	pric (₹
27	2014	Hyundai	Grand i10 SPORTZ 1.2 KAPPA VTVT	Mumbai	18534	Petrol	2nd Owner	33909
41	2015	Hyundai	Grand i10 SPORTS 1.2 VTVT	Mumbai	9307	Petrol	1st Owner	41769

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	pric (₹
62	2014	Hyundai	Grand i10 ASTA 1.2 KAPPA VTVT	Mumbai	37891	Petrol	2nd Owner	36169
69	2014	Hyundai	Grand i10 SPORTS 1.2 VTVT	Mumbai	42967	Petrol	2nd Owner	36909
70	2014	Hyundai	Grand i10 SPORTS 1.2 VTVT	Mumbai	35944	Petrol	1st Owner	38099
...	...	...	...	...	...	...	...	
3220	2017	Hyundai	Grand i10 MAGNA 1.2 KAPPA VTVT	Ahmedabad	76337	Petrol + CNG	1st Owner	45489
3254	2018	Hyundai	Grand i10 SPORTS 1.2 VTVT	Ahmedabad	37000	Petrol + CNG	1st Owner	48649
3255	2018	Hyundai	Grand i10 SPORTZ O 1.2	Ahmedabad	42498	Petrol	1st Owner	51019
3291	2016	Hyundai	Grand i10 SPORTS 1.2 VTVT	Ahmedabad	20970	Petrol	1st Owner	45000
3330	2015	Hyundai	Grand i10 SPORTS 1.2 VTVT	Kolkata	32515	Petrol	1st Owner	37309

133 rows × 9 columns

In [18]:
```
e=a.loc[(a.ref_model == "Alto")]
e
```

Out[18]:

	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
6	2010	Maruti	Alto K10 VXI	Mumbai	50742	Petrol	2nd Owner	170399
7	2015	Maruti	Alto K10 VXI	Mumbai	12657	Petrol	1st Owner	282299
17	2010	Maruti	Alto LXI	Mumbai	34995	Petrol	1st Owner	165999
34	2015	Maruti	Alto 800 VXI	Mumbai	8322	Petrol	1st Owner	286399
...	...	...	...	...	...	...	...	...
3319	2006	Maruti	Alto LXI	Ahmedabad	58542	Petrol + CNG	2nd Owner	85000
3326	2017	Maruti	Alto 800 LXI	Kolkata	11705	Petrol	1st Owner	283199
3332	2019	Maruti	Alto VXI	Kolkata	1306	Petrol	1st Owner	390999
3333	2018	Maruti	Alto 800 LXI	Kolkata	3697	Petrol	1st Owner	316999

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
3336	2019	Maruti	Alto VXI	Kolkata	2000	Petrol	1st Owner	381299

393 rows × 9 columns

In [19]:
```
t=pd.concat([c,d,e])
t
```

Out[19]:

	model_year	maker	model_name	city	distance_covered (km)	fuel_type	pre_owner	price (₹)
8	2013	Maruti	Wagon R 1.0 LXI	Mumbai	13688	Petrol	1st Owner	326199
14	2011	Maruti	Wagon R 1.0 VXI	Mumbai	18514	Petrol	1st Owner	269399
16	2012	Maruti	Wagon R 1.0 LXI	Mumbai	20712	Petrol	2nd Owner	258399
19	2012	Maruti	Wagon R 1.0 VXI	Mumbai	39652	Petrol	3rd Owner	288299
21	2014	Maruti	Wagon R 1.0 VXI	Mumbai	6858	Petrol	1st Owner	358399
...	...	...	...	...	...	...	...	...
3319	2006	Maruti	Alto LXI	Ahmedabad	58542	Petrol + CNG	2nd Owner	85000
3326	2017	Maruti	Alto 800 LXI	Kolkata	11705	Petrol	1st Owner	283199
3332	2019	Maruti	Alto VXI	Kolkata	1306	Petrol	1st Owner	390999
3333	2018	Maruti	Alto 800 LXI	Kolkata	3697	Petrol	1st Owner	316999
3336	2019	Maruti	Alto VXI	Kolkata	2000	Petrol	1st Owner	381299

760 rows × 9 columns

In [20]:
```
f=t.drop(['maker','city','fuel_type','model_name'],axis=1)
f
```

Out[20]:

	model_year	distance_covered (km)	pre_owner	price (₹)	ref_model
8	2013	13688	1st Owner	326199	Wagon
14	2011	18514	1st Owner	269399	Wagon
16	2012	20712	2nd Owner	258399	Wagon
19	2012	39652	3rd Owner	288299	Wagon
21	2014	6858	1st Owner	358399	Wagon
...	...	...	...	...	...
3319	2006	58542	2nd Owner	85000	Alto

	model_year	distance_covered (km)	pre_owner	price (₹)	ref_model
3326	2017	11705	1st Owner	283199	Alto
3332	2019	1306	1st Owner	390999	Alto
3333	2018	3697	1st Owner	316999	Alto
3336	2019	2000	1st Owner	381299	Alto

760 rows × 5 columns

In [21]:
```
g=pd.get_dummies(f,dtype=int)
g
```

Out[21]:

	model_year	distance_covered (km)	price (₹)	pre_owner_1st Owner	pre_owner_2nd Owner	pre_owner_3rd Owner	pre_own
8	2013	13688	326199	1	0	0	
14	2011	18514	269399	1	0	0	
16	2012	20712	258399	0	1	0	
19	2012	39652	288299	0	0	1	
21	2014	6858	358399	1	0	0	
...	...	...	...	...	...	...	
3319	2006	58542	85000	0	1	0	
3326	2017	11705	283199	1	0	0	
3332	2019	1306	390999	1	0	0	
3333	2018	3697	316999	1	0	0	
3336	2019	2000	381299	1	0	0	

760 rows × 10 columns

In [22]:
```
y=g['price (₹)']
y
```

Out[22]:
```
8       326199
14      269399
16      258399
19      288299
21      358399
         ...
3319     85000
3326    283199
3332    390999
3333    316999
3336    381299
Name: price (₹), Length: 760, dtype: int64
```

In [23]:
```
x=g.drop(['price (₹)'],axis=1)
x
```

Out[23]:

	model_year	distance_covered (km)	pre_owner_1st Owner	pre_owner_2nd Owner	pre_owner_3rd Owner	pre_owner_4th Owner
8	2013	13688	1	0	0	0
14	2011	18514	1	0	0	0
16	2012	20712	0	1	0	0
19	2012	39652	0	0	1	0
21	2014	6858	1	0	0	0
...	...	...	...	...	...	...
3319	2006	58542	0	1	0	0
3326	2017	11705	1	0	0	0
3332	2019	1306	1	0	0	0
3333	2018	3697	1	0	0	0
3336	2019	2000	1	0	0	0

760 rows × 9 columns

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=40)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
ypred=reg.predict(x_test)
ypred
```

Out[26]:
```
array([283770.61538551, 170238.22642647, 365858.7279661 , 382955.10963196,
       232377.44230667, 252076.03991584, 468863.16390803, 282017.99924774,
       200061.09482659, 419733.93562396, 269005.0432395 , 211565.48036998,
       213027.59963018, 364230.65501588, 345406.58209312, 317647.5867049 ,
       237814.33296242, 445709.30678488, 284497.86434187, 290184.06564435,
       209486.47685545, 391795.39675611, 320307.50051371, 448544.36885468,
       323494.64340363, 330698.77067152, 430504.03923678, 391586.1823099 ,
       264542.54053135, 332450.98140791, 399691.14716179, 117764.34467441,
       229555.06555849, 285253.14986449, 386469.73431858, 480633.61534459,
       262659.80051443, 222585.35898333, 298999.61924662, 409518.15626961,
       230335.69804841, 222102.57125721, 426513.35727744, 406049.23484375,
       208572.68921538, 340668.36648594, 280051.5865563 , 328629.55482979,
       423238.76249813, 430188.6267417 , 305060.14800056, 285335.32887571,
       233696.58123219, 251624.01357258, 304556.67016532, 322850.45111805,
       248255.83042859, 378468.27832059, 216480.66855795, 459290.50750811,
       402285.73477752, 263424.64336515, 256360.6365695 , 319402.3934289 ,
       172982.13740824, 317155.64705208, 321724.87677395, 389309.50117259,
       255283.70711116, 237466.87602356, 441913.97289348, 451356.7969358 ,
       379827.65641312, 304062.18245092, 274362.18817907, 266611.0044476 ,
       397331.94141345, 240891.87627861, 262427.78412145, 360399.30315534,
       485509.20458179, 223962.54845648, 219873.61422954, 215871.32332452,
       294143.53289343, 280610.63932294, 319662.62625431, 211135.90248558,
```

```
480046.92214387, 268538.84974931, 283133.94942322, 344432.43774103,
373324.38294205, 301034.82560982, 262380.55110934, 291782.22051911,
253267.98558794, 304917.75603916, 360617.75347225, 462618.51570299,
281043.26875765, 321607.68490684, 225906.94317923, 334314.62394221,
214396.70652404, 298846.87631166, 435885.16408487, 208782.02369366,
362828.10194424, 299692.84377532, 340042.94718452, 310426.84715707,
440228.57586123, 346463.25983688, 253081.98511619, 247915.05912825,
209809.35281492, 349277.52412377, 282919.13144818, 238450.7553292 ,
413207.9951873 , 252110.74320614, 210011.1327862 , 295422.62488719,
426332.25416447, 384982.31431881, 386789.76454271, 187298.97612154,
297279.14338844, 347798.13568234, 169680.69016037, 284356.38301691,
262375.9122952 , 164612.99783975, 275711.80712181, 190074.35331348,
135509.53236746, 385783.36601494, 461389.6400663 , 343465.81971225,
186559.47712696, 406478.90377074, 247685.90036242, 293002.01019915,
280674.96301494, 234864.72293916, 232983.28091604, 183480.96031542,
346970.58372751, 313138.21245786, 325759.23719531, 237231.2901476 ,
312036.29661872, 400080.81572407, 251978.75323923, 222507.17290048,
237288.64547063, 259854.37858868, 356914.33354563, 210175.01464398,
232692.20580485, 264212.89803043, 358043.5072547 , 269197.85817874,
165320.12789209, 230158.56981979, 314760.52397988, 483546.9624442 ,
405277.98907914, 273266.01436511, 362308.97654375, 205023.50277586,
190090.3118033 , 177684.10983412, 351722.35831799, 419405.26661839,
300293.45743256, 469301.80491183, 238010.08643175, 474878.95980166,
325465.370427  , 328264.77064402, 368555.31009995, 220616.3683483 ,
255167.21216699, 348765.46556695, 308522.95188187, 286601.65066332,
285209.35487339,  50445.07999306, 295550.23368166, 241389.00790665,
324445.03650983, 348815.76282696, 321958.40487425, 214530.75736152,
328334.21331269, 306608.14937714, 337328.56556185, 183922.60270822,
328823.81861744, 205043.76855285, 202702.54520055, 278937.32173197,
383750.13083868, 345723.03094246, 270471.12724441, 288398.01123317,
184448.25722312, 291394.76936775, 319904.00517637, 239809.43034658,
344216.70912954, 208690.75872578,  89390.87804759, 381050.62821877,
235374.83450527, 296326.72549973, 280691.44957708, 200966.47848381,
442337.44337298, 412319.46036077, 333931.81160498, 223841.26693847,
220146.12216572, 322136.2300258 , 226442.93371855, 284370.1077877 ,
294695.18026136, 429037.27325807, 282051.09808981, 469114.00121122,
454823.73839405, 457120.86293383, 288021.28346359, 323349.26809726,
305125.17651987, 255934.24560393, 244796.89064669, 222430.99666727,
226854.85108276, 235712.36082111, 297157.15001462, 306642.48335411,
241075.94780367, 320028.72336677, 174276.20425683, 449452.96451952,
442960.80489875, 302394.58490749, 464843.60158648])
```

In [27]:
```
from sklearn.metrics import r2_score
r2_score(ypred,y_test)
```

Out[27]:
```
0.799186149472823
```

In [28]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(ypred,y_test)
```

Out[28]:
```
1394087036.3226497
```

In [29]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(5)
```

Out[29]:

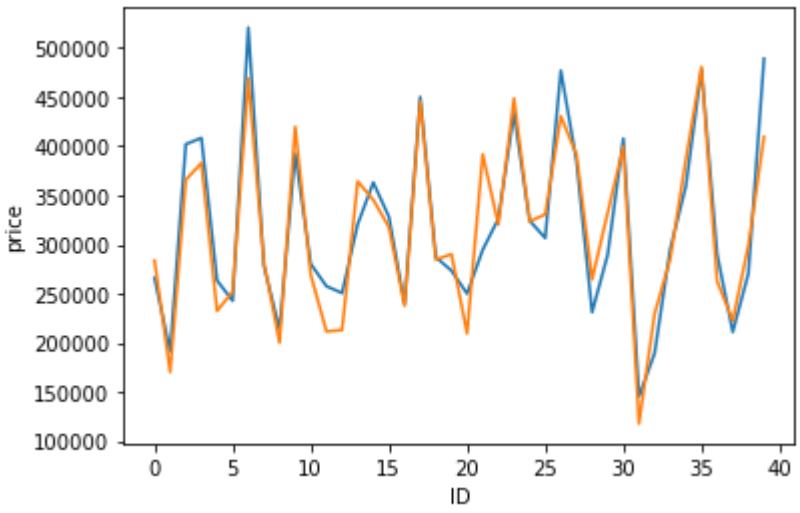
	index	price	predicted	ID
0	1999	265999	283770.615386	0
1	1231	191399	170238.226426	1
2	2523	401999	365858.727966	2
3	2970	408399	382955.109632	3
4	1024	263399	232377.442307	4

In [30]:

```
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[30]:

```
[]
```



elastic regression

In [31]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet
elastic=ElasticNet()
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}
regg=GridSearchCV(elastic,parameters)
regg.fit(x_train,y_train)
```

Out[31]:

```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In [32]:

```
regg.best_params_
```

Out[32]:

```
{'alpha': 0.001}
```

In [33]:

```
elastic=ElasticNet(alpha=0.001)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [34]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[34]:
```
0.8214991608886456
```

In [35]:
```
elastic_error=mean_squared_error(y_test,y_pred_elastic)
elastic_error
```

Out[35]:
```
1393694806.3799038
```

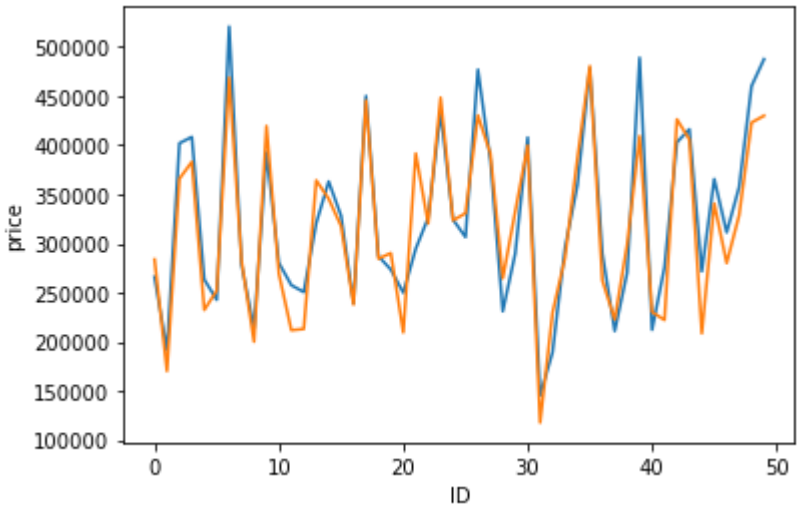
In [36]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=y_pred_elastic
results=results.reset_index()
results["ID"]=results.index
results.head(5)
```

Out[36]:

	index	price	predicted	ID
0	1999	265999	283839.650145	0
1	1231	191399	170254.849489	1
2	2523	401999	365893.239935	2
3	2970	408399	383065.185998	3
4	1024	263399	232417.332477	4

In [37]:
```
sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()
```

Out[37]:
```
[]
```



ridge regression

In [38]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
```

```
ridge=Ridge()
parameters={'alpha':alpha}
regressor=GridSearchCV(ridge,parameters)
regressor.fit(x_train,y_train)
```

Out[38]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [39]:
```
regressor.best_params_
```

Out[39]:
```
{'alpha': 0.01}
```

In [40]:
```
ridge=Ridge(0.01)
ridge.fit(x_train,y_train)
y_pred=ridge.predict(x_test)
y_pred
```

Out[40]:
```
array([283773.33858249, 170238.88047155, 365860.08335093, 382959.45061637,
       232379.01632082, 252077.94052882, 468855.10706502, 282020.76098604,
       200062.61261588, 419725.4395979 , 269007.33135676, 211574.2372946 ,
       213029.59916171, 364228.58521692, 345410.2292263 , 317647.72293168,
       237816.54720092, 445700.99932675, 284500.56444641, 290184.04602872,
       209487.78737362, 391786.74823397, 320307.578247  , 448536.75883704,
       323495.4108349 , 330701.98145477, 430495.30636816, 391587.73171455,
       264544.92678235, 332454.91344436, 399693.27101817, 117764.64018754,
       229554.09276303, 285254.75825454, 386472.15602332, 480626.05944624,
       262662.22816828, 222585.30634215, 299002.76023242, 409509.12511052,
       230337.31696202, 222104.36412371, 426504.71216688, 406041.03265473,
       208574.01982842, 340669.50894023, 280054.38443779, 328630.20224088,
       423230.94918393, 430179.90080924, 305063.92259613, 285336.17567299,
       233703.00948703, 251625.92412595, 304557.09427106, 322853.83449166,
       248255.20617505, 378459.17018572, 216481.83236834, 459282.65407524,
       402276.86266474, 263427.05419954, 256363.20274664, 319403.25085178,
       172982.73821238, 317155.79409697, 321725.67602401, 389311.10064314,
       255285.53718516, 237468.33811729, 441905.74889755, 451349.12507078,
       379829.46439646, 304062.61033113, 274364.35848873, 266610.73633578,
       397323.17823826, 240894.02283964, 262430.21687751, 360403.38037305,
       485501.54146562, 223964.30752038, 219882.18845215, 215869.89875877,
       294146.02798226, 280620.91703574, 319666.07263087, 211137.17673165,
       480039.37914734, 268538.53924274, 283136.68662096, 344436.10629637,
       373326.33393721, 301037.92893961, 262379.62334239, 291782.16575886,
       253267.25821307, 304918.93198995, 360619.22411001, 462610.59618443,
       281043.45015491, 321608.49383372, 225908.65948445, 334318.50789609,
       214398.66884822, 298846.66619439, 435877.06556725, 208782.59701735,
       362819.33774845, 299692.6150546 , 340046.712268  , 310427.14217325,
       440220.38182877, 346464.27485717, 253083.86360769, 247914.4494682 ,
       209810.6562328 , 349281.08613206, 282921.87336992, 238452.19578669,
       413199.63557184, 252110.80106549, 210019.92389209, 295425.85163344,
       426324.36572242, 384984.0089474 , 386780.46631252, 187300.0147737 ,
       297279.72753238, 347801.73022349, 169681.36356577, 284359.08623274,
       262378.33909228, 164613.78268746, 275712.09866218, 190082.82305832,
       135516.92267636, 385774.85680147, 461381.74757162, 343466.90774814,
       186559.77935532, 406469.93944706, 247687.89751782, 293001.92861483,
       280675.15251151, 234866.24225616, 232984.84160735, 183481.33024266,
       346974.95625272, 313141.8094108 , 325760.71461207, 237232.75742204,
       312039.15801802, 400071.98499941, 251978.05421553, 222508.96396917,
       237290.87126938, 259853.49927446, 356915.12583873, 210176.31712038,
       232693.77289711, 264215.28443082, 358045.03450204, 269197.54027976,
       165320.89008981, 230160.1855289 , 314760.72369531, 483539.34247918,
       405279.99717641, 273266.36678995, 362313.01176634, 205023.83633292,
       190091.28907191, 177692.092264  , 351723.26478648, 419396.77072033,
```

```
300293.96818983, 469294.49820838, 238011.53657988, 474871.53045245,
325466.08742084, 328265.43317665, 368556.60618495, 220618.1938975 ,
255169.0448028 , 348769.03883579, 308525.89054222, 286601.70272797,
285212.44382064,  50448.36837823, 295552.6978361 , 241391.14353539,
324448.38481734, 348819.33498973, 321961.80786462, 214531.9640519 ,
328334.87431823, 306608.52836932, 337332.38323692, 183922.96292342,
328827.07063228, 205045.17676947, 202703.25221648, 278937.54234089,
383754.45434   , 345724.06934061, 270473.38312139, 288400.6255706 ,
184449.35856469, 291394.72312783, 319907.45334451, 239810.8409258 ,
344217.78065282, 208692.09384207,  89386.90928172, 381055.01108415,
235376.34260455, 296329.92526446, 280694.24048719, 200967.22367714,
442329.21006463, 412318.61241014, 333932.34951518, 223842.26888385,
220147.96515566, 322139.62910566, 226445.39802254, 284369.4562397 ,
294697.66321906, 429029.32533064, 282053.85910024, 469105.93885211,
454815.99028842, 457113.06431273, 288021.31140914, 323350.03872544,
305128.18280013, 255936.82115772, 244798.95133356, 222432.02962554,
226856.5394431 , 235713.85439824, 297159.57883177, 306645.46336736,
241077.33053123, 320029.5599165 , 174277.52928939, 449445.33452118,
442952.55788221, 302395.04945941, 464835.63313664])
```

In [41]:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[41]:

```
0.8214509579063864
```

In [42]:

```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=y_pred
results=results.reset_index()
results.head(10)
```

Out[42]:

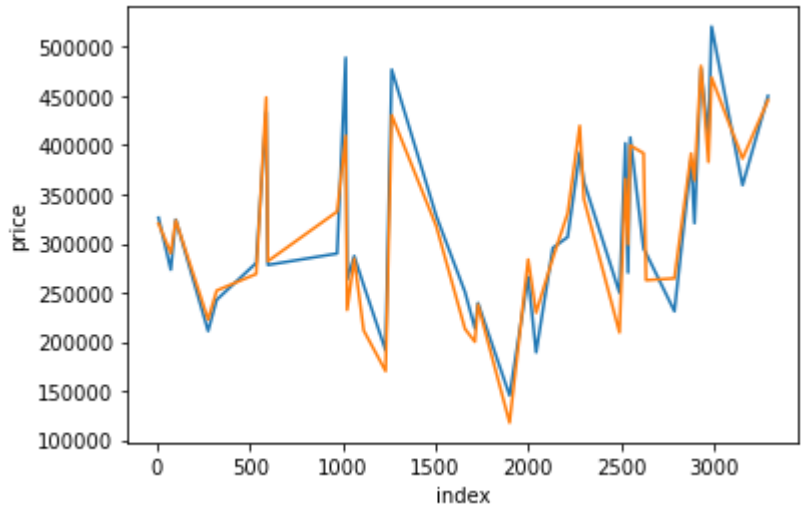
	index	price	predicted
0	1999	265999	283773.338582
1	1231	191399	170238.880472
2	2523	401999	365860.083351
3	2970	408399	382959.450616
4	1024	263399	232379.016321
5	321	242699	252077.940529
6	2987	520699	468855.107065
7	597	278199	282020.760986
8	1712	213799	200062.612616
9	2277	392699	419725.439598

In [43]:

```
sns.lineplot(x='index',y='price',data=results.head(40))
sns.lineplot(x='index',y='predicted',data=results.head(40))
plt.plot()
```

Out[43]:

```
[]
```



lasso regression

In [44]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
lasso=Lasso()
parameters={'alpha':alpha}
regressor=GridSearchCV(lasso,parameters)
regressor.fit(x_train,y_train)
```

Out[44]:
```
GridSearchCV(estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [45]:
```
regressor.best_params_
```

Out[45]:
```
{'alpha': 1e-08}
```

In [46]:
```
lasso=Lasso(1e-08)
lasso.fit(x_train,y_train)
y_pred=lasso.predict(x_test)
y_pred
```

Out[46]:
```
array([283770.61538552, 170238.22642643, 365858.72796615, 382955.10963199,
       232377.44230668, 252076.03991584, 468863.16390797, 282017.99924774,
       200061.09482654, 419733.93562388, 269005.04323951, 211565.48037031,
       213027.59963018, 364230.65501612, 345406.58209315, 317647.58670494,
       237814.33296242, 445709.30678481, 284497.86434185, 290184.06564439,
       209486.47685542, 391795.39675601, 320307.50051375, 448544.36885461,
       323494.64340368, 330698.77067155, 430504.03923671, 391586.18230996,
       264542.54053136, 332450.98140793, 399691.14716181, 117764.34467439,
       229555.06555849, 285253.14986452, 386469.73431864, 480633.61534454,
       262659.80051444, 222585.35898335, 298999.6192466 , 409518.15626954,
       230335.69804841, 222102.57125718, 426513.35727737, 406049.23484364,
       208572.68921535, 340668.36648595, 280051.58655628, 328629.55482981,
       423238.76249805, 430188.62674163, 305060.14800057, 285335.32887574,
       233696.58123254, 251624.01357258, 304556.67016536, 322850.45111807,
       248255.83042858, 378468.27832051, 216480.66855794, 459290.50750802,
       402285.73477744, 263424.64336516, 256360.63656951, 319402.39342894,
       172982.13740823, 317155.64705212, 321724.87677395, 389309.50117265,
       255283.70711116, 237466.87602356, 441913.97289341, 451356.79693574,
       379827.65641318, 304062.18245092, 274362.18817908, 266611.00444759,
```

```
       397331.94141338, 240891.87627861, 262427.78412145, 360399.30315536,
       485509.20458174, 223962.54845648, 219873.61422988, 215871.32332454,
       294143.53289344, 280610.63932328, 319662.62625431, 211135.90248555,
       480046.92214382, 268538.84974931, 283133.94942323, 344432.43774106,
       373324.3829421 , 301034.82560983, 262380.55110937, 291782.22051915,
       253267.98558796, 304917.75603919, 360617.75347231, 462618.51570293,
       281043.26875769, 321607.68490689, 225906.94317923, 334314.62394221,
       214396.70652401, 298846.8763117 , 435885.16408477, 208782.02369365,
       362828.10194415, 299692.84377535, 340042.94718455, 310426.84715711,
       440228.57586114, 346463.2598369 , 253081.98511619, 247915.05912827,
       209809.35281488, 349277.5241238 , 282919.13144819, 238450.7553292 ,
       413207.9951872 , 252110.74320617, 210011.13278653, 295422.62488721,
       426332.25416438, 384982.31431886, 386789.7645426 , 187298.9761215 ,
       297279.14338847, 347798.13568237, 169680.69016036, 284356.38301689,
       262375.91229519, 164612.99783973, 275711.80712181, 190074.3533138 ,
       135509.53236777, 385783.36601485, 461389.64006624, 343465.8197123 ,
       186559.47712696, 406478.90377066, 247685.90036242, 293002.01019919,
       280674.96301498, 234864.72293916, 232983.28091605, 183480.96031542,
       346970.58372752, 313138.21245787, 325759.23719534, 237231.29014761,
       312036.29661874, 400080.81572396, 251978.75323925, 222507.17290048,
       237288.64547063, 259854.37858867, 356914.33354568, 210175.01464397,
       232692.20580485, 264212.89803041, 358043.50725476, 269197.85817877,
       165320.12789205, 230158.56981976, 314760.52397992, 483546.96244415,
       405277.98907921, 273266.01436514, 362308.97654377, 205023.50277586,
       190090.31180326, 177684.10983443, 351722.35831805, 419405.26661829,
       300293.45743256, 469301.80491177, 238010.08643176, 474878.95980161,
       325465.37042701, 328264.77064407, 368555.31010001, 220616.36834827,
       255167.21216699, 348765.46556698, 308522.95188189, 286601.65066332,
       285812.5249086 ,  50445.07999298, 295550.23368168, 241389.00790665,
       324445.03650986, 348815.76282699, 321958.40487427, 214530.75736151,
       328334.21331274, 306608.14937718, 337328.56556185, 183922.60270822,
       328823.81861746, 205043.76855282, 202702.54520054, 278937.32173197,
       383750.13083871, 345723.03094251, 270471.12724442, 288398.01123316,
       184448.25722308, 291394.76936778, 319904.00517639, 239809.43034659,
       344216.70912959, 208690.75872578,  89390.87804754, 381050.62821879,
       235374.83450527, 296326.72549973, 280691.44957709, 200966.4784838 ,
       442337.44337291, 412319.46036103, 333931.81160504, 223841.26693847,
       220146.12216572, 322136.23002582, 226442.93371855, 284370.10778773,
       294695.18026137, 429037.27325797, 282051.09808982, 469114.00121116,
       454823.73839399, 457120.86293377, 288021.28346363, 323349.2680973 ,
       305125.17651986, 255934.24560394, 244796.8906467 , 222430.99666727,
       226854.85108273, 235712.36082108, 297157.15001463, 306642.48335413,
       241075.94780368, 320028.72336677, 174276.20425678, 449452.96451945,
       442960.80489868, 302394.58490749, 464843.60158642])
```

In [47]:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[47]:

```
0.8214652300119228
```

In [48]:

```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=y_pred
results=results.reset_index()
results.head(10)
```

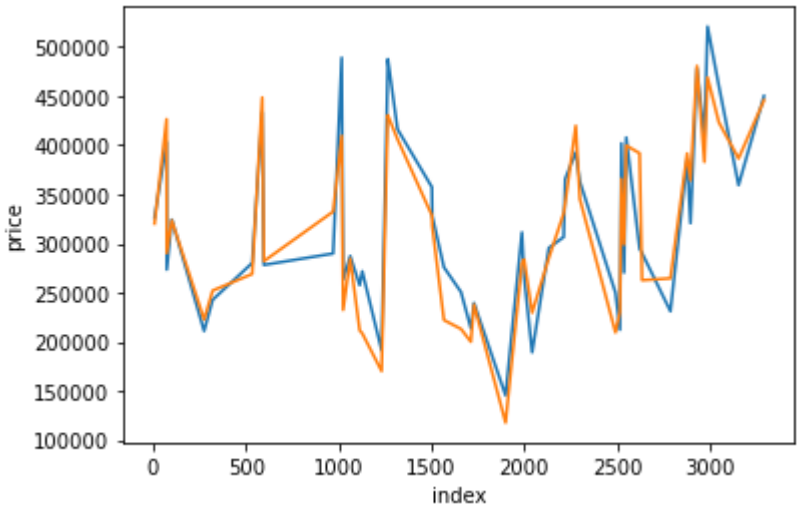
Out[48]:

	index	price	predicted
0	1999	265999	283770.615386
1	1231	191399	170238.226426

	index	price	predicted
2	2523	401999	365858.727966
3	2970	408399	382955.109632
4	1024	263399	232377.442307
5	321	242699	252076.039916
6	2987	520699	468863.163908
7	597	278199	282017.999248
8	1712	213799	200061.094827
9	2277	392699	419733.935624

In [49]:
```
sns.lineplot(x='index',y='price',data=results.head(50))#orange=real,blue=predicted
sns.lineplot(x='index',y='predicted',data=results.head(50))
plt.plot()
```

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



In [50]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['mse']#mean square error
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[50]:
```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse'], 'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [51]:
```
rfc_reg.best_params_
```

Out[51]:
```
{'criterion': 'mse', 'max_depth': 5, 'n_estimators': 25}
```

In [58]:
```
reg=RandomForestRegressor(n_estimators=25,criterion='mse',max_depth=5)
```

In [53]:
```
reg.fit(x_train,y_train)
```

Out[53]:
```
RandomForestRegressor(max_depth=5, n_estimators=175)
```

In [54]:
```
ypred=reg.predict(x_test)
ypred
```

Out[54]:
```
array([271753.52112605, 184438.1393591 , 379696.02123859, 351853.60999179,
       233825.49568796, 245272.62097702, 494811.27188632, 270665.7892115 ,
       219158.38671403, 399363.57041506, 259275.33250317, 224503.51892052,
       220197.3926187 , 338228.28699   , 335731.33261234, 325043.48618892,
       234856.93113709, 451022.99819392, 277137.95175514, 283024.09531198,
       233088.03868907, 332067.00969687, 325982.57311988, 413046.86862096,
       329970.99614533, 320340.7165261 , 439784.03417516, 421734.19339263,
       259924.80558251, 314417.43900092, 410203.41642753, 161304.45799973,
       193564.08150652, 295343.03057253, 401597.12669923, 515062.95745194,
       259243.66642806, 235622.79838154, 299554.30176583, 432705.21835337,
       225328.50760933, 221342.36657104, 448633.17737878, 365645.96380551,
       229439.49723269, 342708.01580599, 269695.13999038, 338673.49517918,
       419472.96098633, 439941.81683694, 293289.47254337, 291281.97143251,
       236491.28618625, 245272.62097702, 317526.7876541 , 303274.10448401,
       236677.50263647, 357616.39987425, 235428.80867186, 492719.87567531,
       372215.51022215, 259924.80558251, 256239.11223543, 330105.65090723,
       180784.10012687, 325043.48618892, 336647.60025899, 421041.10767834,
       249627.15588303, 242219.32593798, 446031.01125166, 418019.45272733,
       401780.44836725, 324517.72536657, 264133.74527435, 281635.44415008,
       360501.92027092, 235657.94912376, 259070.45613835, 325284.8529841 ,
       522546.13297113, 219697.85363316, 243692.60109096, 176271.10278052,
       283333.55047595, 294775.95480886, 307635.61866457, 236321.05827731,
       513966.65666551, 282022.01625306, 271753.52112605, 333469.55968077,
       397310.23073481, 296755.60694467, 236490.18007373, 283776.38010726,
       236902.64549361, 295343.03057253, 368701.07742598, 492186.61953711,
       282951.54958815, 329970.99614533, 221275.87673862, 321831.13731418,
       219530.94921734, 284940.56935372, 445075.04118716, 226460.0528818 ,
       341376.96074541, 284940.56935372, 321426.94276996, 325054.88792467,
       451352.64440501, 345119.17314559, 245652.63515657, 237056.07507514,
       233159.59845964, 351509.4029879 , 271753.52112605, 243656.91481645,
       378586.79082593, 273978.70868441, 224075.03079874, 294582.7202263 ,
       392478.1012838 , 409010.99818428, 401080.41665249, 209479.71793547,
       314807.51174654, 350794.49544381, 178101.88144656, 277137.95175514,
       257086.22131164, 178244.0243037 , 282156.63324768, 219372.64225739,
       174952.52602245, 344295.05180768, 480245.95821042, 340145.86817256,
       176742.19428936, 413827.80111552, 238272.11412677, 283941.93222847,
       282951.54958815, 237332.68949224, 235532.55454754, 179464.20782586,
       314417.43900092, 295205.44834805, 302286.55630557, 238268.59461827,
       316802.34970359, 401111.5657611 , 236716.42904339, 219487.94381194,
       234856.93113709, 245843.07081125, 345855.47620599, 220647.50528126,
       233990.00518477, 257593.11809876, 364874.32971878, 282430.36049056,
       180271.92896827, 238017.02252478, 325043.48618892, 511389.31652779,
       415459.60685139, 274283.11690797, 334017.28704097, 235249.54123868,
       211621.03763119, 211999.06097738, 345910.29753561, 441916.34421271,
       316632.91266558, 437534.4912969 , 242548.11013657, 482313.2304034 ,
       337093.78908675, 336936.13689391, 380590.2661819 , 221095.6825276 ,
       248147.22572278, 350788.85353905, 307746.09528205, 283147.015888  ,
       287868.140128  , 175929.85874354, 283512.36940582, 236507.38234875,
       304746.62268386, 350788.85353905, 300202.1627674 , 229999.05164901,
       336936.13689391, 317526.7876541 , 325383.73690753, 177642.0901036 ,
       317247.12250589, 220577.22376866, 215434.74221225, 282031.3887737 ,
       353716.97820252, 340978.64427044, 260340.53303506, 281437.03925969,
       209223.19382784, 283616.00122448, 298274.69223874, 245012.80972715,
       340740.61198209, 220510.73393623, 122769.75401715, 349003.18009608,
```

```
       237673.76350518, 297148.13534766, 270665.7892115 , 214992.67980593,
       445717.66543557, 399302.83157926, 338962.0318141 , 244189.46859579,
       219530.94921734, 300387.75500572, 235402.75549083, 291292.77765964,
       283333.55047595, 396534.82824731, 270665.7892115 , 494811.27188632,
       435597.83541049, 469485.41023139, 283024.09531198, 329970.99614533,
       314390.68795306, 256239.11223543, 237018.82047861, 244189.46859579,
       234230.65551157, 245528.66780371, 283250.5050191 , 304645.49658025,
       245012.80972715, 335140.44666687, 208991.148802  , 414247.45272733,
       446116.12711358, 322977.71388449, 490479.75914049])
```

In [55]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[55]:
```
0.850043144241902
```

In [56]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=y_pred
results=results.reset_index()
results.head(10)
```

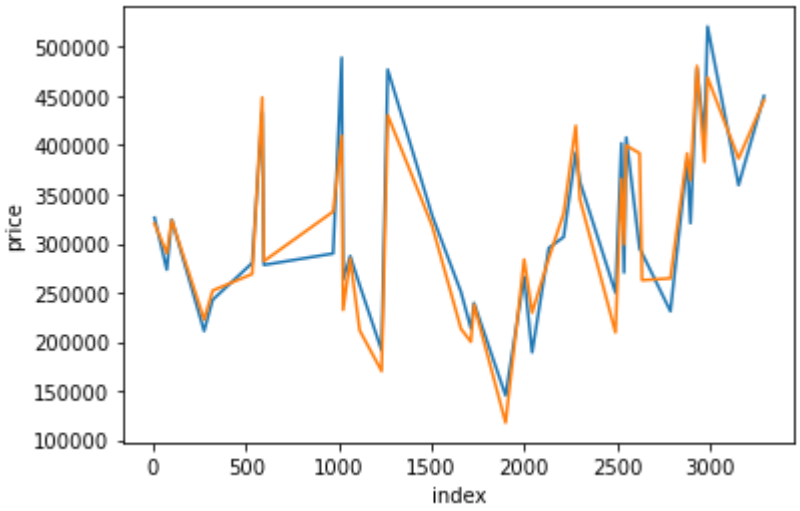
Out[56]:

	index	price	predicted
0	1999	265999	283770.615386
1	1231	191399	170238.226426
2	2523	401999	365858.727966
3	2970	408399	382955.109632
4	1024	263399	232377.442307
5	321	242699	252076.039916
6	2987	520699	468863.163908
7	597	278199	282017.999248
8	1712	213799	200061.094827
9	2277	392699	419733.935624

In [60]:
```
sns.lineplot(x='index',y='price',data=results.head(40))#orange=real,blue=predicted

sns.lineplot(x='index',y='predicted',data=results.head(40))
plt.plot()
```

Out[60]:
```
[]
```


500000
450000
400000
350000
300000
250000
200000
150000
100000
price
0
500
1000
1500
2000
2500
3000
index

In [ ]:

In [ ]: