Import Libraries

In [76]:

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

%matplotlib inline
sns.set()
```

In [77]:

```
conda install -c https://conda.anaconda.org/plotly plotly
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated p
ackages.
```

load dataset

In [78]:

```
# reading data with pandas
df = pd.read_csv('Car_details.csv')
```

In [79]:

```
# detect how many rows and columns
df.shape
```

Out[79]:

```
(8127, 13)
```

In [80]:

```
df.head()
```

Out[80]:

	name	year	selling_price	km_driven	fuel	seller_type	trans
0	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	
1	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	
2	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	
3	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	
4	Hyundai Xcent 1.2 VTVT E Plus	2017	440000	45000	Petrol	Individual	

Explore data analysis

In [81]:

```
#here we can see the Dtype of each column and check if th
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8127 entries, 0 to 8126
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           8127 non-null   object 
 1   year           8127 non-null   int64  
 2   selling_price  8127 non-null   int64  
 3   km_driven      8127 non-null   int64  
 4   fuel           8127 non-null   object 
 5   seller_type    8127 non-null   object 
 6   transmission   8127 non-null   object 
 7   owner          8127 non-null   object 
 8   mileage        7906 non-null   object 
 9   engine         7906 non-null   object 
 10  max_power      7912 non-null   object 
 11  torque         7905 non-null   object 
 12  seats          7906 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.5+ KB
```

In [82]:

```
#displays summary statistics
df.describe()
```

Out[82]:

	year	selling_price	km_driven	seats
count	8127.000000	8.127000e+03	8.127000e+03	7906.000000
mean	2013.803987	6.382950e+05	6.981020e+04	5.416772
std	4.044497	8.063003e+05	5.654780e+04	0.959637
min	1983.000000	2.999900e+04	1.000000e+00	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	14.000000

In [83]:

```
df.columns
```

Out[83]:

```
Index(['name', 'year', 'selling_price', 'km_driven', 'fue
l', 'seller_type',
       'transmission', 'owner', 'mileage', 'engine', 'max
_power', 'torque',
       'seats'],
      dtype='object')
```

In [84]:

```
#show the data types for each columns
df.dtypes
```

Out[84]:

```
name               object
year                int64
selling_price       int64
km_driven           int64
fuel               object
seller_type        object
transmission       object
owner              object
mileage            object
engine             object
max_power          object
torque             object
seats             float64
dtype: object
```

In [85]:

```
# check how many null values in our dataset
df.isna()
```

Out[85]:

	name	year	selling_price	km_driven	fuel	seller_type	transr
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
8122	False	False	False	False	False	False	
8123	False	False	False	False	False	False	
8124	False	False	False	False	False	False	
8125	False	False	False	False	False	False	
8126	False	False	False	False	False	False	

8127 rows × 13 columns

In [86]:

```
#here we can see that the last 5 cloumn has null values
df.isna().sum()
```

Out[86]:

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```

In [87]:

```
#we can see the percentage null values from the datasest
df.isna().sum()/raw_data.shape[0]
```

Out[87]:

```
name             0.000000
year             0.000000
selling_price    0.000000
km_driven        0.000000
fuel             0.000000
seller_type      0.000000
transmission     0.000000
owner            0.000000
mileage          0.027193
engine           0.027193
max_power        0.026455
torque           0.027316
seats            0.027193
dtype: float64
```

Cleaning

In [88]:

```
#we gonna fill the seats column with mean(float64)
df.seats.fillna(df.seats.mean(),inplace=True)
```

In [89]:

```
df.seats.mean()
```

Out[89]:

```
5.416772071844173
```

In [90]:

```
#we gonna fill the engine column with mode(object)
df.engine.fillna(df.engine.mode()[0],inplace=True)
```

In [91]:

```
df.engine.mode()
```

Out[91]:

```
0    1248 CC
dtype: object
```

In [92]:

```
#we gonna fill the engine column with mode(object)
df.mileage.fillna(df.mileage.mode()[0],inplace=True)#we g
```

In [93]:

```
df.mileage.mode()
```

Out[93]:

```
0    18.9 kmpl
dtype: object
```

In [94]:

```
# drop the'torque','max_power'
df.drop(columns=['torque','max_power'],inplace=True)
```

In [95]:

```
#now we can see there is no more empty valuse
df.isna().sum()
```

Out[95]:

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
mileage          0
engine           0
seats            0
dtype: int64
```

In [96]:

```
#cheack if there is duplicated and we found 1202
df.duplicated().sum()
```

Out[96]:

```
1202
```

In [97]:

```
#drop the duplicated
df.drop_duplicates(inplace=True)
```

In [98]:

```
#notice that after we clean our dataset the num of the co
df.shape
```

Out[98]:

```
(6925, 11)
```

visualization

In [99]:

```
#count of selling prices in every year
# plotting the bar chart
fig = px.bar(df,x="year", y='selling_price')

# showing the plot
fig.show()
```

In [100…

```
 #count of selling prices according to it's km_driven
# plotting the histogram
fig = px.histogram(df, x="km_driven", y="selling_price")

# showing the plot
fig.show()
```

In [101…

```
 #sum of selling_price for each cars
# plotting the bar chart
fig = px.histogram(df,x="name", y='selling_price')

# showing the plot
fig.show()
```

In [102…

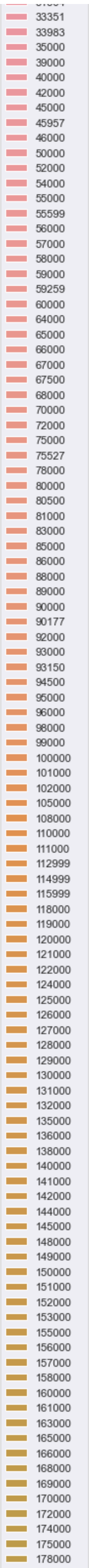
```
#count of selling prices in every year using countplot
plt.figure(figsize=[15,8])
sns.countplot(df.year,hue='selling_price',data=df)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decor
ators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From ver
sion 0.12, the only valid positional argument will be `da
ta`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
```

Out[102…

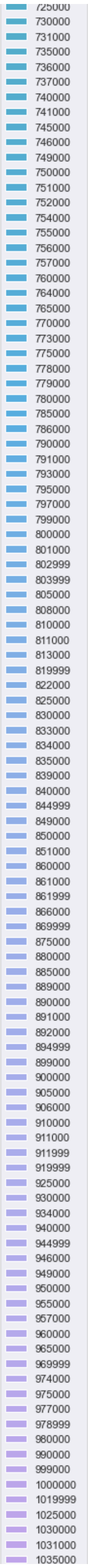
```
<AxesSubplot:xlabel='year', ylabel='count'>
```


33351
33983
35000
39000
40000
42000
45000
45957
46000
50000
52000
54000
55000
55599
56000
57000
58000
59000
59259
60000
64000
65000
66000
67000
67500
68000
70000
72000
75000
75527
78000
80000
80500
81000
83000
85000
86000
88000
89000
90000
90177
92000
93000
93150
94500
95000
96000
98000
99000
100000
101000
102000
105000
108000
110000
111000
112999
114999
115999
118000
119000
120000
121000
122000
124000
125000
126000
127000
128000
129000
130000
131000
132000
135000
136000
138000
140000
141000
142000
144000
145000
148000
149000
150000
151000
152000
153000
155000
156000
157000
158000
160000
161000
163000
165000
166000
168000
169000
170000
172000
174000
175000
178000

179000
180000
181000
184000
185000
187000
189000
190000
191000
194000
195000
196000
198000
199000
200000
202999
204000
204999
206000
209000
210000
211000
212000
215000
217000
219000
220000
221000
222000
225000
227000
227999
229999
231000
235000
236000
238000
240000
241000
242000
244000
245000
246000
248000
250000
250999
252000
252999
254000
254999
256000
256999
258000
259000
260000
261000
262000
265000
267000
268000
270000
272000
275000
276000
277000
278000
279000
280000
282000
285000
287000
290000
291000
292000
295000
297000
298000
299000
300000
302000
305000
307000
308000
310000
311000
312000
313000
315000
316000
317000
318000
319000
320000
321000
325000
327000
328000
329000
330000
335000
337000
339000
340000
341000

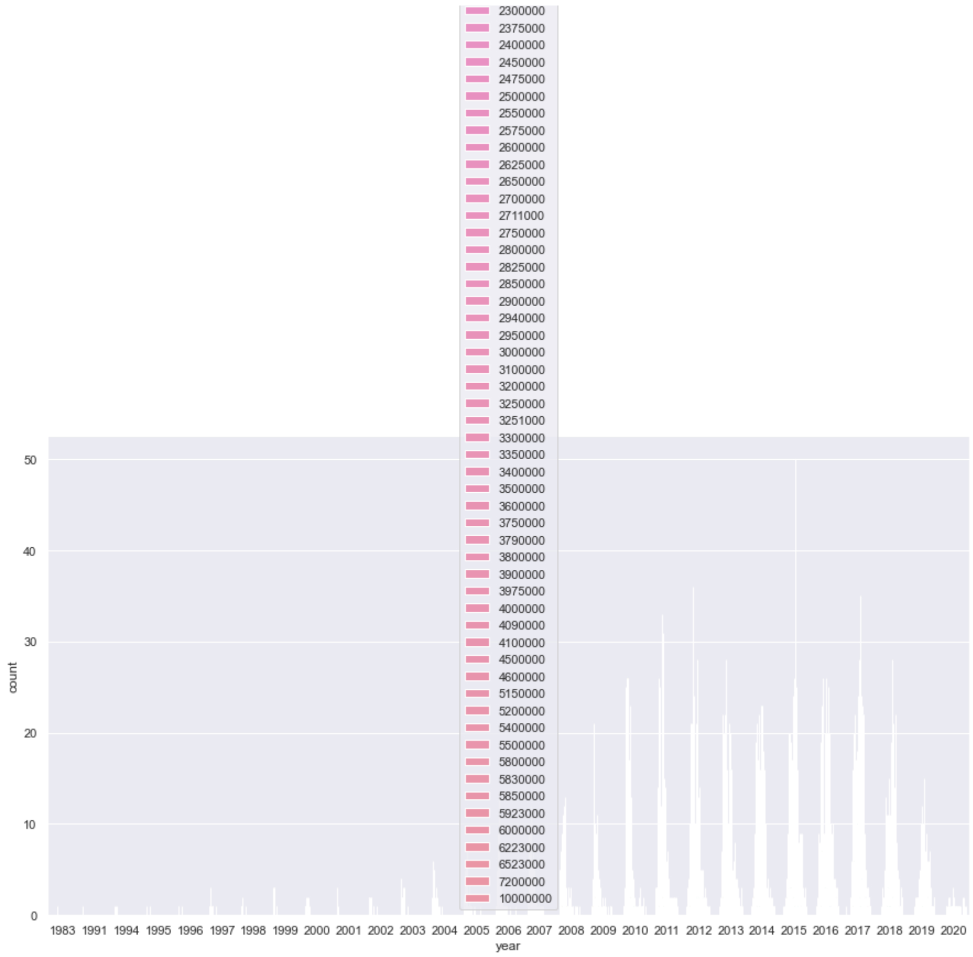
341000
345000
346000
347000
348000
350000
351000
352000
355000
356000
357000
358000
359000
360000
361000
362000
365000
366000
369000
370000
371000
372000
374000
375000
376000
377000
378000
380000
382000
385000
386000
387000
388000
389000
390000
391000
392000
395000
399000
400000
401000
403000
404000
405000
408000
409000
409999
411000
412000
415000
416000
420000
421000
423000
425000
426000
426999
428000
429000
430000
430999
432000
433000
434000
434999
438999
440000
441000
444000
445000
448000
449000
450000
451000
451999
453000
454000
455000
455999
458000
459999
465000
466000
467000
469000
470000
472000
475000
476999
479000
480000
482000
483000
484999
486000
487000
488000
488999
490000
493000
495000
497000
499000
500000

501000
503000
509999
511000
512000
515000
516000
520000
521000
522000
524000
525000
526000
528000
530000
531000
532000
535000
537000
539000
540000
541000
545000
549000
550000
551000
555000
556000
558000
560000
561000
562000
563000
565000
567000
570000
571000
573000
574000
575000
576000
577000
578000
579000
580000
585000
589000
590000
591000
595000
597000
599000
600000
604000
605000
610000
611000
615000
616000
620000
624000
625000
626000
630000
631000
632000
634000
635000
640000
643000
645000
649000
650000
651000
654000
655000
660000
665000
666000
667000
669000
670000
671000
675000
678000
679000
680000
681000
684000
685000
690000
693000
694000
695000
698000
699000
700000
710000
711000
715000
720000
721000
722000



725000
730000
731000
735000
736000
737000
740000
741000
745000
746000
749000
750000
751000
752000
754000
755000
756000
757000
760000
764000
765000
770000
773000
775000
778000
779000
780000
785000
786000
790000
791000
793000
795000
797000
799000
800000
801000
802999
803999
805000
808000
810000
811000
813000
819999
822000
825000
830000
833000
834000
835000
839000
840000
844999
849000
850000
851000
860000
861000
861999
866000
869999
875000
880000
885000
889000
890000
891000
892000
894999
899000
900000
905000
906000
910000
911000
911999
919999
925000
930000
934000
940000
944999
946000
949000
950000
955000
957000
960000
965000
969999
974000
975000
977000
978999
980000
990000
999000
1000000
1019999
1025000
1030000
1031000
1035000

1040000
1044999
1050000
1051000
1075000
1080000
1085000
1090000
1100000
1110000
1125000
1132000
1140000
1143000
1145000
1147000
1149000
1150000
1151000
1160000
1165000
1175000
1180000
1190000
1200000
1210000
1220000
1225000
1227000
1237000
1250000
1251000
1265000
1270000
1275000
1282000
1290000
1295000
1300000
1325000
1350000
1365000
1380000
1385000
1390000
1400000
1405000
1425000
1445000
1450000
1460000
1465000
1475000
1490000
1500000
1511000
1515000
1516000
1520000
1525000
1530000
1550000
1560000
1575000
1576000
1590000
1594000
1600000
1625000
1630000
1650000
1670000
1675000
1680000
1689999
1700000
1745000
1748999
1750000
1757000
1789999
1800000
1825000
1850000
1859000
1864999
1888000
1898999
1900000
1920000
1925000
1938000
1950000
2000000
2051000
2064000
2100000
2125000
2150000
2175000
2199000
2200000
2280000



In [103…

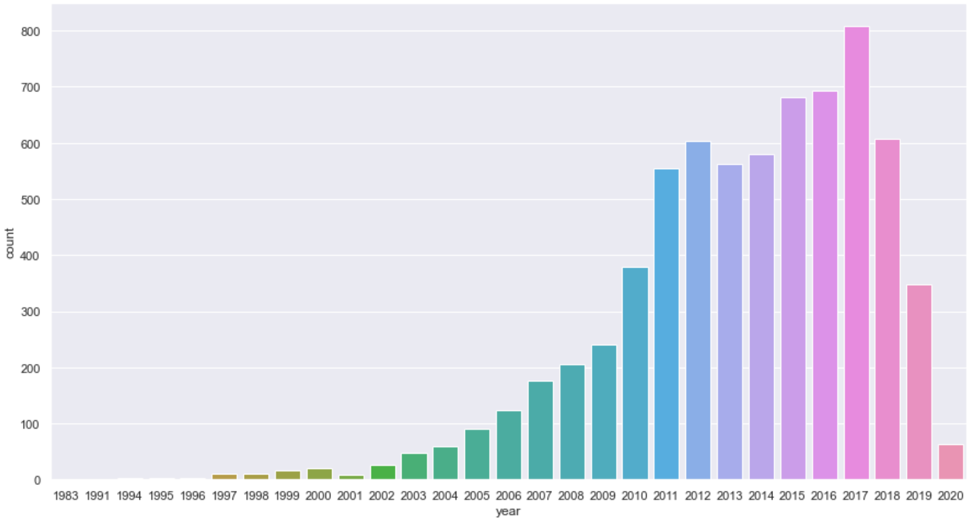
```
# plotting the countplot for the yaer
plt.figure(figsize=[15,8])
sns.countplot(df.year)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decor
ators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From ver
sion 0.12, the only valid positional argument will be `da
ta`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
```

Out[103…

```
<AxesSubplot:xlabel='year', ylabel='count'>
```



In [104…

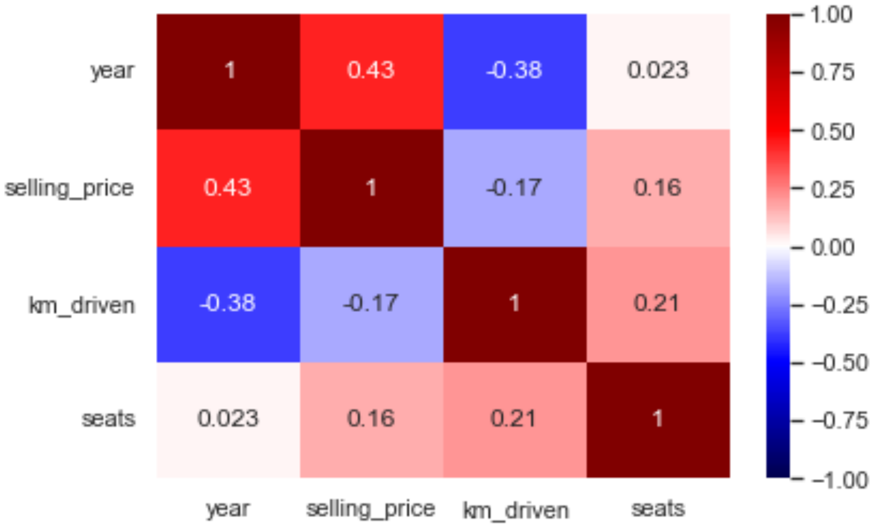
```
#find the correlation between variables
df.corr()
```

Out[104…

	year	selling_price	km_driven	seats
year	1.000000	0.433080	-0.377070	0.023098
selling_price	0.433080	1.000000	-0.165615	0.157154
km_driven	-0.377070	-0.165615	1.000000	0.205931
seats	0.023098	0.157154	0.205931	1.000000

In [105…

```
# visualize the correlations between the variables
sns.heatmap(df.corr(), cmap="seismic", annot=True, vmin=-
```



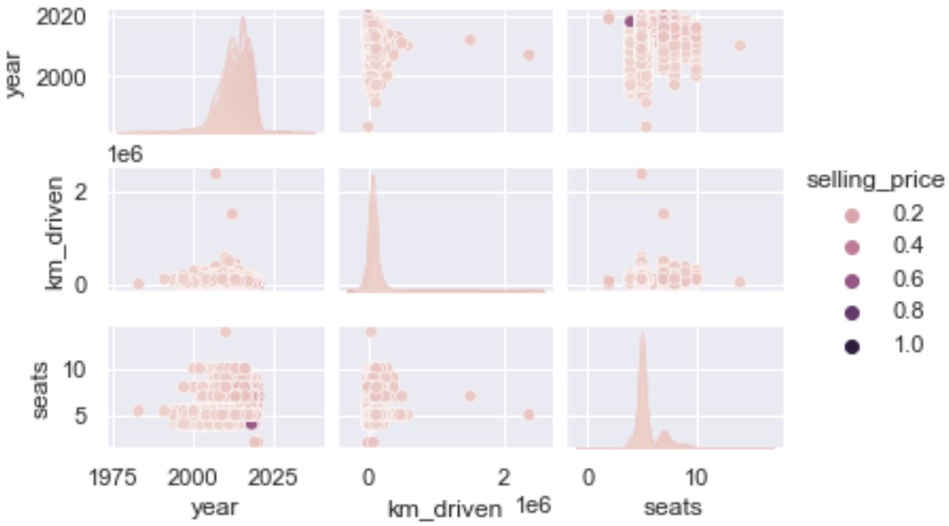
In [106…

```
#to visualize the relationship between each variable
plt.figure(figsize=[20,30])
sns.pairplot(df,diag_kind='kde',hue='selling_price',heigh
```

Out[106…

```
<seaborn.axisgrid.PairGrid at 0x7ff8823005b0>
```

```
<Figure size 1440x2160 with 0 Axes>
```



preprocessing for modeling

In [107…

```
#convert character columns to dummy variables to be ready
df=pd.get_dummies(df)
```

In [108…

```
df.head()
```

Out[108…

	year	selling_price	km_driven	seats	name_Ambassador CLASSIC 1500 DSL AC	name_A Classic
0	2014	370000	120000	5.0	0	
1	2006	158000	140000	5.0	0	
2	2010	225000	127000	5.0	0	
3	2007	130000	120000	5.0	0	
4	2017	440000	45000	5.0	0	

5 rows × 2590 columns

In [109…

```
#you can see that the columns increase after dummies vari
df.shape
```

Out[109…

```
(6925, 2590)
```

split Data

In [110…

```
# detect input and output
x=df.drop('selling_price',axis=1)
y=df.selling_price
y
```

Out[110…

```
0       370000
1       158000
2       225000
3       130000
4       440000
         ...
8120    260000
8121    475000
8122    320000
8123    135000
8124    382000
Name: selling_price, Length: 6925, dtype: int64
```

In [111…

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y,
```

In [112…

```
X_train
```

Out[112…

	year	km_driven	seats	name_Ambassador CLASSIC 1500 DSL AC	name_Ambassador Classic 2000 DSZ AC PS
2464	2008	58632	5.0	0	0
2437	2011	50000	5.0	0	0
7515	2018	50000	5.0	0	0
3063	2006	62900	5.0	0	0
1236	2013	300000	8.0	0	0
...	...	...	...	...	...
5701	1997	100000	5.0	0	0
3620	2008	110000	5.0	0	0
1817	2013	70000	9.0	0	0
2844	2017	90000	7.0	0	0
2981	2016	59300	5.0	0	0

5540 rows × 2589 columns

In [113…

```
y_train
```

Out[113…

```
2464      114999
2437      250000
7515     5500000
3063      135000
1236      200000
          ...
5701       70000
3620      250000
1817      500000
2844     1050000
2981      240000
Name: selling_price, Length: 5540, dtype: int64
```

In [114…

```
X_test
```

Out[114…

	year	km_driven	seats	name_Ambassador CLASSIC 1500 DSL AC	name_Ambassador Classic 2000 DSZ AC PS
1173	2012	70000	5.0	0	0
942	2008	100000	5.0	0	0
4282	2016	94000	5.0	0	0
313	2014	80100	5.0	0	0
5397	2011	25000	5.0	0	0
...	...	...	...	...	...
4104	2008	136500	8.0	0	0
351	2012	40000	5.0	0	0
2575	2019	14000	5.0	0	0
3350	2017	44000	5.0	0	0
4274	2016	120000	5.0	0	0

1385 rows × 2589 columns

In [115…

```
y_test
```

Out[115…

```
1173     254999
942      290000
4282    1100000
313      555000
5397     245000
          ...
4104     375000
351      325000
2575     380000
3350    2600000
4274     350000
Name: selling_price, Length: 1385, dtype: int64
```

In [116…

```
# feature scaling
#scale all the futur to same scale
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Modeling

In [117…

```
# apply LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [118…

```
linreg_model = LinearRegression()
linreg_model.fit(x,y)
```

Out[118…

```
LinearRegression()
```

In [119…

```
y_pred = linreg_model.predict(x)
print(metrics.mean_absolute_error(y, y_pred))
```

```
44834.248419519965
```

In [120…

```
print(linreg_model.intercept_)
print(linreg_model.coef_)
```

```
3405192073097.6074
[ 3.07951709e+04 -2.07833909e-01 -9.01974974e+04 ... -3.7
7938069e+11
 -4.43547773e+11 -1.81815062e+11]
```

Model evaluation

In [121…

```
# R-Squared
linreg_model.score(x, y)
```

Out[121…

```
0.9741883150866637
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js