In [149…
```
import pandas as pd
import numpy as np
```

a) Creating and loading different datasets in Python.

In [150…
```
df=pd.DataFrame({'Number':[1,2,3,4,5,6,7,8,9,10],'Squares':[1,4,9,16,25,36,49,64,81
```

In [151…
```
df
```

Out[151]:

	Number	Squares
0	1	1
1	2	4
2	3	9
3	4	16
4	5	25
5	6	36
6	7	49
7	8	64
8	9	81
9	10	100

In [152…
```
df.to_csv("num_and_sqrs.csv",index=False)
```

In [153…
```
a1=pd.read_csv('num_and_sqrs.csv')
```

In [154…
```
a1
```

Out[154]:

	Number	Squares
0	1	1
1	2	4
2	3	9
3	4	16
4	5	25
5	6	36
6	7	49
7	8	64
8	9	81
9	10	100

In [155…
```
print(a1)
```

```
   Number  Squares
0       1        1
1       2        4
2       3        9
3       4       16
4       5       25
5       6       36
6       7       49
7       8       64
8       9       81
9      10      100
```

In [156…
```
################################################################################
```

Cars_scraping dataset operations

In [157…
```
import pandas as pd
import numpy as np
```

In [158…
```
df=pd.read_csv('cars_scrap_missing.csv')
```

In [159…
```
df
```

Out[159]:

	name	year	miles	condition	price
0	Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0
1	Chevrolet Silverado 1500	2021	15138.0	1 accident reported, 1 Owner	38008.0
2	Toyota RAV4	2022	32879.0	No accidents reported, 1 Owner	24988.0
3	Honda Civic	2020	37190.0	No accidents reported, 1 Owner	18998.0
4	Honda Civic	2020	27496.0	No accidents reported, 1 Owner	NaN
...	...	...	...	...	...
94	Nissan Versa	2017	87358.0	1 accident reported, 1 Owner	8488.0
95	Jeep Wrangler	2021	19914.0	No accidents reported, 1 Owner	35998.0
96	Ford Edge	2017	89932.0	No accidents reported, 2 Owners	13998.0
97	Toyota Camry	2023	19140.0	No accidents reported, 1 Owner	24575.0
98	BMW 7 Series	2019	66896.0	No accidents reported, 1 Owner	28624.0

99 rows × 5 columns

In [160…
```
df.tail(10)
```

Out[160]:

	name	year	miles	condition	price
89	Volvo XC90	2016	126749.0	No accidents reported, 3 Owners	19998.0
90	Honda Odyssey	2021	30252.0	No accidents reported, 1 Owner	35498.0
91	Kia Forte	2021	37385.0	No accidents reported, 1 Owner	14999.0
92	Lexus RX	2020	37965.0	No accidents reported, 1 Owner	35998.0
93	Mercedes-Benz E-Class	2022	27642.0	No accidents reported, 1 Owner	41998.0
94	Nissan Versa	2017	87358.0	1 accident reported, 1 Owner	8488.0
95	Jeep Wrangler	2021	19914.0	No accidents reported, 1 Owner	35998.0
96	Ford Edge	2017	89932.0	No accidents reported, 2 Owners	13998.0
97	Toyota Camry	2023	19140.0	No accidents reported, 1 Owner	24575.0
98	BMW 7 Series	2019	66896.0	No accidents reported, 1 Owner	28624.0

In [161…

```
df1=df.head(50)
df1
```

Out[161]:

	name	year	miles	condition	price
0	Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0
1	Chevrolet Silverado 1500	2021	15138.0	1 accident reported, 1 Owner	38008.0
2	Toyota RAV4	2022	32879.0	No accidents reported, 1 Owner	24988.0
3	Honda Civic	2020	37190.0	No accidents reported, 1 Owner	18998.0
4	Honda Civic	2020	27496.0	No accidents reported, 1 Owner	NaN
5	Honda Civic	2020	29026.0	1 accident reported, 1 Owner	21000.0
6	Honda Accord	2019	NaN	1 accident reported, 1 Owner	20998.0
7	Mercedes-Benz GLC	2018	57602.0	1 accident reported, 3 Owners	18887.0
8	Honda Civic	2020	50240.0	1 accident reported, 1 Owner	19498.0
9	BMW 5 Series	2013	56766.0	No accidents reported, 3 Owners	17998.0
10	Jeep Wrangler	2018	NaN	No accidents reported, 2 Owners	30998.0
11	Honda Civic	2020	11140.0	No accidents reported, 1 Owner	20998.0
12	Honda Civic	2020	8506.0	No accidents reported, 1 Owner	NaN
13	Mercedes-Benz GLC	2021	38760.0	No accidents reported, 1 Owner	29998.0
14	Porsche Macan	2020	NaN	No accidents reported, 2 Owners	39998.0
15	Honda Civic	2021	22850.0	No accidents reported, 1 Owner	21998.0
16	Buick Cascada	2016	68660.0	1 accident reported, 2 Owners	17998.0
17	BMW 3 Series	2014	55079.0	No accidents reported, 2 Owners	17998.0
18	Jeep Grand Cherokee	2017	64337.0	No accidents reported, 2 Owners	19598.0
19	Mercedes-Benz C-Class	2018	NaN	1 accident reported, 2 Owners	18285.0
20	Porsche Cayenne	2021	21991.0	No accidents reported, 1 Owner	NaN
21	Honda Odyssey	2019	61318.0	No accidents reported, 1 Owner	26998.0
22	Mercedes-Benz GLC	2020	36064.0	1 accident reported, 1 Owner	26839.0
23	BMW 7 Series	2015	NaN	No accidents reported, 3 Owners	NaN
24	Mercedes-Benz GLC	2020	NaN	1 accident reported, 1 Owner	NaN
25	Kia K5	2022	33632.0	No accidents reported, 1 Owner	19988.0
26	Nissan Kicks	2021	51560.0	No accidents reported, 1 Owner	16488.0
27	Mercedes-Benz C-Class	2021	NaN	No accidents reported, 1 Owner	29990.0
28	Honda CR-V	2020	74031.0	No accidents reported, 1 Owner	23998.0
29	Honda Civic	2020	16112.0	No accidents reported, 1 Owner	20998.0
30	Honda CR-V	2019	17123.0	1 accident reported, 1 Owner	NaN
31	Honda Civic	2016	71406.0	1 accident reported, 2 Owners	16498.0
32	Chrysler Pacifica	2018	73428.0	No accidents reported, 2 Owners	19998.0
33	Volkswagen Tiguan	2020	NaN	1 accident reported, 1 Owner	20998.0
34	BMW 3 Series	2015	60478.0	1 accident reported, 2 Owners	16998.0
35	Porsche Cayenne	2020	44880.0	No accidents reported, 3 Owners	NaN

	name	year	miles	condition	price
36	Honda CR-V	2022	23215.0	No accidents reported, 1 Owner	27798.0
37	Nissan Sentra	2021	24044.0	No accidents reported, 1 Owner	20998.0
38	Volkswagen Tiguan	2019	50950.0	No accidents reported, 1 Owner	20998.0
39	Toyota Camry	2020	NaN	1 accident reported, 1 Owner	23998.0
40	Chevrolet Malibu	2022	32668.0	No accidents reported, 1 Owner	NaN
41	Volkswagen Tiguan	2021	50142.0	No accidents reported, 1 Owner	17998.0
42	Chevrolet Volt	2017	91969.0	1 accident reported, 1 Owner	14998.0
43	Land Rover Range Rover Sport	2016	NaN	1 accident reported, 2 Owners	20994.0
44	Toyota Highlander	2021	16462.0	No accidents reported, 1 Owner	NaN
45	Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0
46	Kia Optima	2015	NaN	No accidents reported, 1 Owner	15998.0
47	Dodge Charger	2016	120296.0	No accidents reported, 1 Owner	NaN
48	Jeep Grand Cherokee	2015	29063.0	No accidents reported, 1 Owner	23998.0
49	Toyota Corolla	2022	33167.0	No accidents reported, 1 Owner	20222.0

In [162…
```
df1.shape
```

Out[162]:
```
(50, 5)
```

In [163…
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   name       50 non-null     object 
 1   year       50 non-null     int64  
 2   miles      39 non-null     float64
 3   condition  50 non-null     object 
 4   price      40 non-null     float64
dtypes: float64(2), int64(1), object(2)
memory usage: 2.1+ KB
```

In [164…
```
df1.describe()
```

Out[164]:

	year	miles	price
count	50.000000	39.000000	40.000000
mean	2019.120000	43396.923077	21950.275000
std	2.412891	23738.501082	5689.936514
min	2013.000000	8506.000000	14998.000000
25%	2018.000000	25770.000000	17998.000000
50%	2020.000000	38760.000000	20996.000000
75%	2021.000000	57184.000000	23998.000000
max	2022.000000	120296.000000	39998.000000

In [165…

```
df1.isna()  # checks for missing or null values(NA,NaN)
```

Out[165]:

	name	year	miles	condition	price
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	True
5	False	False	False	False	False
6	False	False	True	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	False	True	False	False
11	False	False	False	False	False
12	False	False	False	False	True
13	False	False	False	False	False
14	False	False	True	False	False
15	False	False	False	False	False
16	False	False	False	False	False
17	False	False	False	False	False
18	False	False	False	False	False
19	False	False	True	False	False
20	False	False	False	False	True
21	False	False	False	False	False
22	False	False	False	False	False
23	False	False	True	False	True
24	False	False	True	False	True
25	False	False	False	False	False
26	False	False	False	False	False
27	False	False	True	False	False
28	False	False	False	False	False
29	False	False	False	False	False
30	False	False	False	False	True
31	False	False	False	False	False
32	False	False	False	False	False
33	False	False	True	False	False
34	False	False	False	False	False
35	False	False	False	False	True

	name	year	miles	condition	price
36	False	False	False	False	False
37	False	False	False	False	False
38	False	False	False	False	False
39	False	False	True	False	False
40	False	False	False	False	True
41	False	False	False	False	False
42	False	False	False	False	False
43	False	False	True	False	False
44	False	False	False	False	True
45	False	False	False	False	False
46	False	False	True	False	False
47	False	False	False	False	True
48	False	False	False	False	False
49	False	False	False	False	False

In [166…

```
df1.isnull()
```

Out[166]:

	name	year	miles	condition	price
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	True
5	False	False	False	False	False
6	False	False	True	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	False	True	False	False
11	False	False	False	False	False
12	False	False	False	False	True
13	False	False	False	False	False
14	False	False	True	False	False
15	False	False	False	False	False
16	False	False	False	False	False
17	False	False	False	False	False
18	False	False	False	False	False
19	False	False	True	False	False
20	False	False	False	False	True
21	False	False	False	False	False
22	False	False	False	False	False
23	False	False	True	False	True
24	False	False	True	False	True
25	False	False	False	False	False
26	False	False	False	False	False
27	False	False	True	False	False
28	False	False	False	False	False
29	False	False	False	False	False
30	False	False	False	False	True
31	False	False	False	False	False
32	False	False	False	False	False
33	False	False	True	False	False
34	False	False	False	False	False
35	False	False	False	False	True

	name	year	miles	condition	price
36	False	False	False	False	False
37	False	False	False	False	False
38	False	False	False	False	False
39	False	False	True	False	False
40	False	False	False	False	True
41	False	False	False	False	False
42	False	False	False	False	False
43	False	False	True	False	False
44	False	False	False	False	True
45	False	False	False	False	False
46	False	False	True	False	False
47	False	False	False	False	True
48	False	False	False	False	False
49	False	False	False	False	False

In [167…
```
df1.isnull().sum()
```

Out[167]:
```
name          0
year          0
miles        11
condition     0
price        10
dtype: int64
```

In [168…
```
df1['name']=='Honda Civic'
```

Out[168]:
```
0     False
1     False
2     False
3      True
4      True
5      True
6     False
7     False
8      True
9     False
10    False
11     True
12     True
13    False
14    False
15     True
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29     True
30    False
31     True
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42    False
43    False
44    False
45    False
46    False
47    False
48    False
49    False
Name: name, dtype: bool
```

In [169…
```
df1.iloc[4:48,:5]
```

Out[169]:

	name	year	miles	condition	price
4	Honda Civic	2020	27496.0	No accidents reported, 1 Owner	NaN
5	Honda Civic	2020	29026.0	1 accident reported, 1 Owner	21000.0
6	Honda Accord	2019	NaN	1 accident reported, 1 Owner	20998.0
7	Mercedes-Benz GLC	2018	57602.0	1 accident reported, 3 Owners	18887.0
8	Honda Civic	2020	50240.0	1 accident reported, 1 Owner	19498.0
9	BMW 5 Series	2013	56766.0	No accidents reported, 3 Owners	17998.0
10	Jeep Wrangler	2018	NaN	No accidents reported, 2 Owners	30998.0
11	Honda Civic	2020	11140.0	No accidents reported, 1 Owner	20998.0
12	Honda Civic	2020	8506.0	No accidents reported, 1 Owner	NaN
13	Mercedes-Benz GLC	2021	38760.0	No accidents reported, 1 Owner	29998.0
14	Porsche Macan	2020	NaN	No accidents reported, 2 Owners	39998.0
15	Honda Civic	2021	22850.0	No accidents reported, 1 Owner	21998.0
16	Buick Cascada	2016	68660.0	1 accident reported, 2 Owners	17998.0
17	BMW 3 Series	2014	55079.0	No accidents reported, 2 Owners	17998.0
18	Jeep Grand Cherokee	2017	64337.0	No accidents reported, 2 Owners	19598.0
19	Mercedes-Benz C-Class	2018	NaN	1 accident reported, 2 Owners	18285.0
20	Porsche Cayenne	2021	21991.0	No accidents reported, 1 Owner	NaN
21	Honda Odyssey	2019	61318.0	No accidents reported, 1 Owner	26998.0
22	Mercedes-Benz GLC	2020	36064.0	1 accident reported, 1 Owner	26839.0
23	BMW 7 Series	2015	NaN	No accidents reported, 3 Owners	NaN
24	Mercedes-Benz GLC	2020	NaN	1 accident reported, 1 Owner	NaN
25	Kia K5	2022	33632.0	No accidents reported, 1 Owner	19988.0
26	Nissan Kicks	2021	51560.0	No accidents reported, 1 Owner	16488.0
27	Mercedes-Benz C-Class	2021	NaN	No accidents reported, 1 Owner	29990.0
28	Honda CR-V	2020	74031.0	No accidents reported, 1 Owner	23998.0
29	Honda Civic	2020	16112.0	No accidents reported, 1 Owner	20998.0
30	Honda CR-V	2019	17123.0	1 accident reported, 1 Owner	NaN
31	Honda Civic	2016	71406.0	1 accident reported, 2 Owners	16498.0
32	Chrysler Pacifica	2018	73428.0	No accidents reported, 2 Owners	19998.0
33	Volkswagen Tiguan	2020	NaN	1 accident reported, 1 Owner	20998.0
34	BMW 3 Series	2015	60478.0	1 accident reported, 2 Owners	16998.0
35	Porsche Cayenne	2020	44880.0	No accidents reported, 3 Owners	NaN
36	Honda CR-V	2022	23215.0	No accidents reported, 1 Owner	27798.0
37	Nissan Sentra	2021	24044.0	No accidents reported, 1 Owner	20998.0
38	Volkswagen Tiguan	2019	50950.0	No accidents reported, 1 Owner	20998.0
39	Toyota Camry	2020	NaN	1 accident reported, 1 Owner	23998.0

	name	year	miles	condition	price
40	Chevrolet Malibu	2022	32668.0	No accidents reported, 1 Owner	NaN
41	Volkswagen Tiguan	2021	50142.0	No accidents reported, 1 Owner	17998.0
42	Chevrolet Volt	2017	91969.0	1 accident reported, 1 Owner	14998.0
43	Land Rover Range Rover Sport	2016	NaN	1 accident reported, 2 Owners	20994.0
44	Toyota Highlander	2021	16462.0	No accidents reported, 1 Owner	NaN
45	Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0
46	Kia Optima	2015	NaN	No accidents reported, 1 Owner	15998.0
47	Dodge Charger	2016	120296.0	No accidents reported, 1 Owner	NaN

In [170…
```
min(df1['price'])
```

Out[170]:
```
14998.0
```

In [171…
```
max(df1['price'])
```

Out[171]:
```
39998.0
```

In [172…
```
df1['price'].mean()
```

Out[172]:
```
21950.275
```

In [173…
```
df1.value_counts() # returns series that contains count of unique values
```

Out[173]:

name	year	miles	condition	price	
Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0	2
BMW 3 Series	2014	55079.0	No accidents reported, 2 Owners	17998.0	1
	2015	60478.0	1 accident reported, 2 Owners	16998.0	1
Volkswagen Tiguan	2019	50950.0	No accidents reported, 1 Owner	20998.0	1
Toyota RAV4	2022	32879.0	No accidents reported, 1 Owner	24988.0	1
Toyota Corolla	2022	33167.0	No accidents reported, 1 Owner	20222.0	1
Nissan Sentra	2021	24044.0	No accidents reported, 1 Owner	20998.0	1
Nissan Kicks	2021	51560.0	No accidents reported, 1 Owner	16488.0	1
Mercedes-Benz GLC	2021	38760.0	No accidents reported, 1 Owner	29998.0	1
	2020	36064.0	1 accident reported, 1 Owner	26839.0	1
	2018	57602.0	1 accident reported, 3 Owners	18887.0	1
Kia K5	2022	33632.0	No accidents reported, 1 Owner	19988.0	1
Jeep Grand Cherokee	2017	64337.0	No accidents reported, 2 Owners	19598.0	1
	2015	29063.0	No accidents reported, 1 Owner	23998.0	1
Honda Odyssey	2019	61318.0	No accidents reported, 1 Owner	26998.0	1
Honda Civic	2021	22850.0	No accidents reported, 1 Owner	21998.0	1
	2020	50240.0	1 accident reported, 1 Owner	19498.0	1
		37190.0	No accidents reported, 1 Owner	18998.0	1
		29026.0	1 accident reported, 1 Owner	21000.0	1
		16112.0	No accidents reported, 1 Owner	20998.0	1
		11140.0	No accidents reported, 1 Owner	20998.0	1
	2016	71406.0	1 accident reported, 2 Owners	16498.0	1
Honda CR-V	2022	23215.0	No accidents reported, 1 Owner	27798.0	1
	2020	74031.0	No accidents reported, 1 Owner	23998.0	1
Chrysler Pacifica	2018	73428.0	No accidents reported, 2 Owners	19998.0	1
Chevrolet Volt	2017	91969.0	1 accident reported, 1 Owner	14998.0	1
Chevrolet Silverado 1500	2021	15138.0	1 accident reported, 1 Owner	38008.0	1
Buick Cascada	2016	68660.0	1 accident reported, 2 Owners	17998.0	1
BMW 5 Series	2013	56766.0	No accidents reported, 3 Owners	17998.0	1
Volkswagen Tiguan	2021	50142.0	No accidents reported, 1 Owner	17998.0	1

Name: count, dtype: int64

b) Reshaping, Filtering, Scaling, Merging the data and Handling the missing values in datasets.

MERGING

In [174…
```
car_name_miles=pd.DataFrame()
car_price=pd.DataFrame()
```

In [175…
```
car_name_miles=pd.read_csv('cars_scrap_missing.csv')

car_name_miles=car_name_miles[['name','miles']]

car_name_miles
```

Out[175]:

	name	miles
0	Kia Forte	41406.0
1	Chevrolet Silverado 1500	15138.0
2	Toyota RAV4	32879.0
3	Honda Civic	37190.0
4	Honda Civic	27496.0
...	...	...
94	Nissan Versa	87358.0
95	Jeep Wrangler	19914.0
96	Ford Edge	89932.0
97	Toyota Camry	19140.0
98	BMW 7 Series	66896.0

99 rows × 2 columns

In [176…
```
car_price=pd.read_csv('cars_scrap_missing.csv')

car_price=car_price[['name','price']]

car_price
```

Out[176]:

	name	price
0	Kia Forte	15988.0
1	Chevrolet Silverado 1500	38008.0
2	Toyota RAV4	24988.0
3	Honda Civic	18998.0
4	Honda Civic	NaN
...	...	...
94	Nissan Versa	8488.0
95	Jeep Wrangler	35998.0
96	Ford Edge	13998.0
97	Toyota Camry	24575.0
98	BMW 7 Series	28624.0

99 rows × 2 columns

In [177…
```
#Merging the dataframe
car_data= car_name_miles.merge(car_price, how='inner',on='name')
```

In [178…
```
car_data
```

Out[178]:

	name	miles	price
0	Kia Forte	41406.0	15988.0
1	Kia Forte	41406.0	15988.0
2	Kia Forte	41406.0	14999.0
3	Kia Forte	41406.0	15988.0
4	Kia Forte	41406.0	15988.0
...	...	...	...
316	Volvo XC90	126749.0	19998.0
317	Lexus RX	37965.0	35998.0
318	Mercedes-Benz E-Class	27642.0	41998.0
319	Nissan Versa	87358.0	8488.0
320	Ford Edge	89932.0	13998.0

321 rows × 3 columns

HANDLING MISSING VALUES

In [179…
```
df1.isnull().sum()
```

Out[179]:
```
name          0
year          0
miles        11
condition     0
price        10
dtype: int64
```

In [180…
```
from sklearn.impute import SimpleImputer
import numpy as np
```

In [181…
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

In [182…
```
df1_handle_NaN = imputer.fit_transform(df1) # Doesn't work as dataset has categoric
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[182], line 1
----> 1 df1_handle_NaN = imputer.fit_transform(df1)

File ~\anaconda3\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return_tuple = (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\anaconda3\Lib\site-packages\sklearn\base.py:915, in TransformerMixin.fit_transform(self, X, y, **fit_params)
    911 # non-optimized default implementation; override when a better
    912 # method is possible for a given clustering algorithm
    913 if y is None:
    914     # fit method of arity 1 (unsupervised transformation)
--> 915     return self.fit(X, **fit_params).transform(X)
    916 else:
    917     # fit method of arity 2 (supervised transformation)
    918     return self.fit(X, y, **fit_params).transform(X)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:1151, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\impute\_base.py:366, in SimpleImputer.fit(self, X, y)
    348 @_fit_context(prefer_skip_nested_validation=True)
    349 def fit(self, X, y=None):
    350     """Fit the imputer on `X`.
    351
    352     Parameters
   (...)
    364         Fitted estimator.
    365     """
--> 366     X = self._validate_input(X, in_fit=True)
    368     # default fill_value is 0 for numerical input and "missing_value"
    369     # otherwise
    370     if self.fill_value is None:

File ~\anaconda3\Lib\site-packages\sklearn\impute\_base.py:327, in SimpleImputer._validate_input(self, X, in_fit)
    321 if "could not convert" in str(ve):
    322     new_ve = ValueError(
    323         "Cannot use {} strategy with non-numeric data:\n{}".format(
    324             self.strategy, ve
    325         )
    326     )
--> 327     raise new_ve from None
    328 else:
    329     raise ve
```

```
ValueError: Cannot use mean strategy with non-numeric data:
could not convert string to float: 'Kia Forte'
```

Label Encoding

In [183…
```
from sklearn.preprocessing import LabelEncoder
```

In [184…
```
df1.head()
```

Out[184]:

	name	year	miles	condition	price
0	Kia Forte	2022	41406.0	No accidents reported, 1 Owner	15988.0
1	Chevrolet Silverado 1500	2021	15138.0	1 accident reported, 1 Owner	38008.0
2	Toyota RAV4	2022	32879.0	No accidents reported, 1 Owner	24988.0
3	Honda Civic	2020	37190.0	No accidents reported, 1 Owner	18998.0
4	Honda Civic	2020	27496.0	No accidents reported, 1 Owner	NaN

In [185…
```
encoder = LabelEncoder()
```

In [186…
```
new_car_data=pd.DataFrame()
new_car_data
```

Out[186]:

In [187…
```
new_car_data['name'] = encoder.fit_transform(df1['name']) # converting car name to
```

In [188…
```
new_car_data
```

Out[188]:

	name
0	15
1	5
2	28
3	11
4	11
5	11
6	9
7	20
8	11
9	1
10	14
11	11
12	11
13	20
14	24
15	11
16	3
17	0
18	13
19	19
20	23
21	12
22	20
23	2
24	20
25	16
26	21
27	19
28	10
29	11
30	10
31	11
32	7
33	29
34	0
35	23

	name
36	10
37	22
38	29
39	25
40	4
41	29
42	6
43	18
44	27
45	15
46	17
47	8
48	13
49	26

In [189…

```
new_car_data['year'] = df1['year']
new_car_data['miles'] = df1['miles']
new_car_data['condition'] = encoder.fit_transform(df1['condition']) # converting co
new_car_data['price'] = df1['price']
```

In [190…

```
new_car_data
```

Out[190]:

	name	year	miles	condition	price
0	15	2022	41406.0	3	15988.0
1	5	2021	15138.0	0	38008.0
2	28	2022	32879.0	3	24988.0
3	11	2020	37190.0	3	18998.0
4	11	2020	27496.0	3	NaN
5	11	2020	29026.0	0	21000.0
6	9	2019	NaN	0	20998.0
7	20	2018	57602.0	2	18887.0
8	11	2020	50240.0	0	19498.0
9	1	2013	56766.0	5	17998.0
10	14	2018	NaN	4	30998.0
11	11	2020	11140.0	3	20998.0
12	11	2020	8506.0	3	NaN
13	20	2021	38760.0	3	29998.0
14	24	2020	NaN	4	39998.0
15	11	2021	22850.0	3	21998.0
16	3	2016	68660.0	1	17998.0
17	0	2014	55079.0	4	17998.0
18	13	2017	64337.0	4	19598.0
19	19	2018	NaN	1	18285.0
20	23	2021	21991.0	3	NaN
21	12	2019	61318.0	3	26998.0
22	20	2020	36064.0	0	26839.0
23	2	2015	NaN	5	NaN
24	20	2020	NaN	0	NaN
25	16	2022	33632.0	3	19988.0
26	21	2021	51560.0	3	16488.0
27	19	2021	NaN	3	29990.0
28	10	2020	74031.0	3	23998.0
29	11	2020	16112.0	3	20998.0
30	10	2019	17123.0	0	NaN
31	11	2016	71406.0	1	16498.0
32	7	2018	73428.0	4	19998.0
33	29	2020	NaN	0	20998.0
34	0	2015	60478.0	1	16998.0
35	23	2020	44880.0	5	NaN

	name	year	miles	condition	price
36	10	2022	23215.0	3	27798.0
37	22	2021	24044.0	3	20998.0
38	29	2019	50950.0	3	20998.0
39	25	2020	NaN	0	23998.0
40	4	2022	32668.0	3	NaN
41	29	2021	50142.0	3	17998.0
42	6	2017	91969.0	0	14998.0
43	18	2016	NaN	1	20994.0
44	27	2021	16462.0	3	NaN
45	15	2022	41406.0	3	15988.0
46	17	2015	NaN	3	15998.0
47	8	2016	120296.0	3	NaN
48	13	2015	29063.0	3	23998.0
49	26	2022	33167.0	3	20222.0

In [191…
```
#Now all column data is in numner now we can handle missing data.
```

In [192…
```
from sklearn.impute import SimpleImputer
import numpy as np
```

In [193…
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

In [194…
```
car_NaN_handle = imputer.fit_transform(new_car_data)
```

In [195…
```
car_NaN_handle
```

Out[195]:
```
array([[1.50000000e+01, 2.02200000e+03, 4.14060000e+04, 3.00000000e+00,
        1.59880000e+04],
       [5.00000000e+00, 2.02100000e+03, 1.51380000e+04, 0.00000000e+00,
        3.80080000e+04],
       [2.80000000e+01, 2.02200000e+03, 3.28790000e+04, 3.00000000e+00,
        2.49880000e+04],
       [1.10000000e+01, 2.02000000e+03, 3.71900000e+04, 3.00000000e+00,
        1.89980000e+04],
       [1.10000000e+01, 2.02000000e+03, 2.74960000e+04, 3.00000000e+00,
        2.19502750e+04],
       [1.10000000e+01, 2.02000000e+03, 2.90260000e+04, 0.00000000e+00,
        2.10000000e+04],
       [9.00000000e+00, 2.01900000e+03, 4.33969231e+04, 0.00000000e+00,
        2.09980000e+04],
       [2.00000000e+01, 2.01800000e+03, 5.76020000e+04, 2.00000000e+00,
        1.88870000e+04],
       [1.10000000e+01, 2.02000000e+03, 5.02400000e+04, 0.00000000e+00,
        1.94980000e+04],
       [1.00000000e+00, 2.01300000e+03, 5.67660000e+04, 5.00000000e+00,
        1.79980000e+04],
       [1.40000000e+01, 2.01800000e+03, 4.33969231e+04, 4.00000000e+00,
        3.09980000e+04],
       [1.10000000e+01, 2.02000000e+03, 1.11400000e+04, 3.00000000e+00,
        2.09980000e+04],
       [1.10000000e+01, 2.02000000e+03, 8.50600000e+03, 3.00000000e+00,
        2.19502750e+04],
       [2.00000000e+01, 2.02100000e+03, 3.87600000e+04, 3.00000000e+00,
        2.99980000e+04],
       [2.40000000e+01, 2.02000000e+03, 4.33969231e+04, 4.00000000e+00,
        3.99980000e+04],
       [1.10000000e+01, 2.02100000e+03, 2.28500000e+04, 3.00000000e+00,
        2.19980000e+04],
       [3.00000000e+00, 2.01600000e+03, 6.86600000e+04, 1.00000000e+00,
        1.79980000e+04],
       [0.00000000e+00, 2.01400000e+03, 5.50790000e+04, 4.00000000e+00,
        1.79980000e+04],
       [1.30000000e+01, 2.01700000e+03, 6.43370000e+04, 4.00000000e+00,
        1.95980000e+04],
       [1.90000000e+01, 2.01800000e+03, 4.33969231e+04, 1.00000000e+00,
        1.82850000e+04],
       [2.30000000e+01, 2.02100000e+03, 2.19910000e+04, 3.00000000e+00,
        2.19502750e+04],
       [1.20000000e+01, 2.01900000e+03, 6.13180000e+04, 3.00000000e+00,
        2.69980000e+04],
       [2.00000000e+01, 2.02000000e+03, 3.60640000e+04, 0.00000000e+00,
        2.68390000e+04],
       [2.00000000e+00, 2.01500000e+03, 4.33969231e+04, 5.00000000e+00,
        2.19502750e+04],
       [2.00000000e+01, 2.02000000e+03, 4.33969231e+04, 0.00000000e+00,
        2.19502750e+04],
       [1.60000000e+01, 2.02200000e+03, 3.36320000e+04, 3.00000000e+00,
        1.99880000e+04],
       [2.10000000e+01, 2.02100000e+03, 5.15600000e+04, 3.00000000e+00,
        1.64880000e+04],
       [1.90000000e+01, 2.02100000e+03, 4.33969231e+04, 3.00000000e+00,
        2.99900000e+04],
       [1.00000000e+01, 2.02000000e+03, 7.40310000e+04, 3.00000000e+00,
        2.39980000e+04],
       [1.10000000e+01, 2.02000000e+03, 1.61120000e+04, 3.00000000e+00,
        2.09980000e+04],
       [1.00000000e+01, 2.01900000e+03, 1.71230000e+04, 0.00000000e+00,
        2.19502750e+04],
       [1.10000000e+01, 2.01600000e+03, 7.14060000e+04, 1.00000000e+00,
        1.64980000e+04],
```

```
       [7.00000000e+00, 2.01800000e+03, 7.34280000e+04, 4.00000000e+00,
        1.99980000e+04],
       [2.90000000e+01, 2.02000000e+03, 4.33969231e+04, 0.00000000e+00,
        2.09980000e+04],
       [0.00000000e+00, 2.01500000e+03, 6.04780000e+04, 1.00000000e+00,
        1.69980000e+04],
       [2.30000000e+01, 2.02000000e+03, 4.48800000e+04, 5.00000000e+00,
        2.19502750e+04],
       [1.00000000e+01, 2.02200000e+03, 2.32150000e+04, 3.00000000e+00,
        2.77980000e+04],
       [2.20000000e+01, 2.02100000e+03, 2.40440000e+04, 3.00000000e+00,
        2.09980000e+04],
       [2.90000000e+01, 2.01900000e+03, 5.09500000e+04, 3.00000000e+00,
        2.09980000e+04],
       [2.50000000e+01, 2.02000000e+03, 4.33969231e+04, 0.00000000e+00,
        2.39980000e+04],
       [4.00000000e+00, 2.02200000e+03, 3.26680000e+04, 3.00000000e+00,
        2.19502750e+04],
       [2.90000000e+01, 2.02100000e+03, 5.01420000e+04, 3.00000000e+00,
        1.79980000e+04],
       [6.00000000e+00, 2.01700000e+03, 9.19690000e+04, 0.00000000e+00,
        1.49980000e+04],
       [1.80000000e+01, 2.01600000e+03, 4.33969231e+04, 1.00000000e+00,
        2.09940000e+04],
       [2.70000000e+01, 2.02100000e+03, 1.64620000e+04, 3.00000000e+00,
        2.19502750e+04],
       [1.50000000e+01, 2.02200000e+03, 4.14060000e+04, 3.00000000e+00,
        1.59880000e+04],
       [1.70000000e+01, 2.01500000e+03, 4.33969231e+04, 3.00000000e+00,
        1.59980000e+04],
       [8.00000000e+00, 2.01600000e+03, 1.20296000e+05, 3.00000000e+00,
        2.19502750e+04],
       [1.30000000e+01, 2.01500000e+03, 2.90630000e+04, 3.00000000e+00,
        2.39980000e+04],
       [2.60000000e+01, 2.02200000e+03, 3.31670000e+04, 3.00000000e+00,
        2.02220000e+04]])
```

In [196…
```
car_NaN_handle=pd.DataFrame(car_NaN_handle)
```

In [197…
```
car_NaN_handle
```

Out[197]:

	0	1	2	3	4
0	15.0	2022.0	41406.000000	3.0	15988.000
1	5.0	2021.0	15138.000000	0.0	38008.000
2	28.0	2022.0	32879.000000	3.0	24988.000
3	11.0	2020.0	37190.000000	3.0	18998.000
4	11.0	2020.0	27496.000000	3.0	21950.275
5	11.0	2020.0	29026.000000	0.0	21000.000
6	9.0	2019.0	43396.923077	0.0	20998.000
7	20.0	2018.0	57602.000000	2.0	18887.000
8	11.0	2020.0	50240.000000	0.0	19498.000
9	1.0	2013.0	56766.000000	5.0	17998.000
10	14.0	2018.0	43396.923077	4.0	30998.000
11	11.0	2020.0	11140.000000	3.0	20998.000
12	11.0	2020.0	8506.000000	3.0	21950.275
13	20.0	2021.0	38760.000000	3.0	29998.000
14	24.0	2020.0	43396.923077	4.0	39998.000
15	11.0	2021.0	22850.000000	3.0	21998.000
16	3.0	2016.0	68660.000000	1.0	17998.000
17	0.0	2014.0	55079.000000	4.0	17998.000
18	13.0	2017.0	64337.000000	4.0	19598.000
19	19.0	2018.0	43396.923077	1.0	18285.000
20	23.0	2021.0	21991.000000	3.0	21950.275
21	12.0	2019.0	61318.000000	3.0	26998.000
22	20.0	2020.0	36064.000000	0.0	26839.000
23	2.0	2015.0	43396.923077	5.0	21950.275
24	20.0	2020.0	43396.923077	0.0	21950.275
25	16.0	2022.0	33632.000000	3.0	19988.000
26	21.0	2021.0	51560.000000	3.0	16488.000
27	19.0	2021.0	43396.923077	3.0	29990.000
28	10.0	2020.0	74031.000000	3.0	23998.000
29	11.0	2020.0	16112.000000	3.0	20998.000
30	10.0	2019.0	17123.000000	0.0	21950.275
31	11.0	2016.0	71406.000000	1.0	16498.000
32	7.0	2018.0	73428.000000	4.0	19998.000
33	29.0	2020.0	43396.923077	0.0	20998.000
34	0.0	2015.0	60478.000000	1.0	16998.000
35	23.0	2020.0	44880.000000	5.0	21950.275

	0	1	2	3	4
36	10.0	2022.0	23215.000000	3.0	27798.000
37	22.0	2021.0	24044.000000	3.0	20998.000
38	29.0	2019.0	50950.000000	3.0	20998.000
39	25.0	2020.0	43396.923077	0.0	23998.000
40	4.0	2022.0	32668.000000	3.0	21950.275
41	29.0	2021.0	50142.000000	3.0	17998.000
42	6.0	2017.0	91969.000000	0.0	14998.000
43	18.0	2016.0	43396.923077	1.0	20994.000
44	27.0	2021.0	16462.000000	3.0	21950.275
45	15.0	2022.0	41406.000000	3.0	15988.000
46	17.0	2015.0	43396.923077	3.0	15998.000
47	8.0	2016.0	120296.000000	3.0	21950.275
48	13.0	2015.0	29063.000000	3.0	23998.000
49	26.0	2022.0	33167.000000	3.0	20222.000

In [198…
```
car_NaN_handle.isnull().sum()
```

Out[198]:
```
0    0
1    0
2    0
3    0
4    0
dtype: int64
```

Creating Dependent and Indepent columns : X and Y

In [199…
```
cardata=pd.DataFrame(car_NaN_handle)
```

In [200…
```
cardata.shape
```

Out[200]:
```
(50, 5)
```

In [201…
```
X = pd.DataFrame(cardata.iloc[:, 0:4].values)
#cardata.iloc[:, 0:4].values extracts the values from the selected rows and columns
```

In [202…
```
X
```

Out[202]:

	0	1	2	3
0	15.0	2022.0	41406.000000	3.0
1	5.0	2021.0	15138.000000	0.0
2	28.0	2022.0	32879.000000	3.0
3	11.0	2020.0	37190.000000	3.0
4	11.0	2020.0	27496.000000	3.0
5	11.0	2020.0	29026.000000	0.0
6	9.0	2019.0	43396.923077	0.0
7	20.0	2018.0	57602.000000	2.0
8	11.0	2020.0	50240.000000	0.0
9	1.0	2013.0	56766.000000	5.0
10	14.0	2018.0	43396.923077	4.0
11	11.0	2020.0	11140.000000	3.0
12	11.0	2020.0	8506.000000	3.0
13	20.0	2021.0	38760.000000	3.0
14	24.0	2020.0	43396.923077	4.0
15	11.0	2021.0	22850.000000	3.0
16	3.0	2016.0	68660.000000	1.0
17	0.0	2014.0	55079.000000	4.0
18	13.0	2017.0	64337.000000	4.0
19	19.0	2018.0	43396.923077	1.0
20	23.0	2021.0	21991.000000	3.0
21	12.0	2019.0	61318.000000	3.0
22	20.0	2020.0	36064.000000	0.0
23	2.0	2015.0	43396.923077	5.0
24	20.0	2020.0	43396.923077	0.0
25	16.0	2022.0	33632.000000	3.0
26	21.0	2021.0	51560.000000	3.0
27	19.0	2021.0	43396.923077	3.0
28	10.0	2020.0	74031.000000	3.0
29	11.0	2020.0	16112.000000	3.0
30	10.0	2019.0	17123.000000	0.0
31	11.0	2016.0	71406.000000	1.0
32	7.0	2018.0	73428.000000	4.0
33	29.0	2020.0	43396.923077	0.0
34	0.0	2015.0	60478.000000	1.0
35	23.0	2020.0	44880.000000	5.0

	0	1	2	3
36	10.0	2022.0	23215.000000	3.0
37	22.0	2021.0	24044.000000	3.0
38	29.0	2019.0	50950.000000	3.0
39	25.0	2020.0	43396.923077	0.0
40	4.0	2022.0	32668.000000	3.0
41	29.0	2021.0	50142.000000	3.0
42	6.0	2017.0	91969.000000	0.0
43	18.0	2016.0	43396.923077	1.0
44	27.0	2021.0	16462.000000	3.0
45	15.0	2022.0	41406.000000	3.0
46	17.0	2015.0	43396.923077	3.0
47	8.0	2016.0	120296.000000	3.0
48	13.0	2015.0	29063.000000	3.0
49	26.0	2022.0	33167.000000	3.0

In [203…
```
Y = pd.DataFrame(cardata.iloc[:, -1].values)
```

In [204…
```
Y = pd.DataFrame(cardata.iloc[:, 4:].values)
```

In [205…
```
Y
```

Out[205]:

	0
0	15988.000
1	38008.000
2	24988.000
3	18998.000
4	21950.275
5	21000.000
6	20998.000
7	18887.000
8	19498.000
9	17998.000
10	30998.000
11	20998.000
12	21950.275
13	29998.000
14	39998.000
15	21998.000
16	17998.000
17	17998.000
18	19598.000
19	18285.000
20	21950.275
21	26998.000
22	26839.000
23	21950.275
24	21950.275
25	19988.000
26	16488.000
27	29990.000
28	23998.000
29	20998.000
30	21950.275
31	16498.000
32	19998.000
33	20998.000
34	16998.000
35	21950.275

	0
36	27798.000
37	20998.000
38	20998.000
39	23998.000
40	21950.275
41	17998.000
42	14998.000
43	20994.000
44	21950.275
45	15988.000
46	15998.000
47	21950.275
48	23998.000
49	20222.000

Feature Scaling of DataSet- MinMaxScalar

In [206…
```
from sklearn.preprocessing import MinMaxScaler
```

In [207…
```
scalar = MinMaxScaler()
```

In [208…
```
X_scaled = pd.DataFrame(scalar.fit_transform(X))
```

In [209…
```
X_scaled
```

Out[209]:

	0	1	2	3
0	0.517241	1.000000	0.294302	0.6
1	0.172414	0.888889	0.059326	0.0
2	0.965517	1.000000	0.218025	0.6
3	0.379310	0.777778	0.256588	0.6
4	0.379310	0.777778	0.169872	0.6
5	0.379310	0.777778	0.183558	0.0
6	0.310345	0.666667	0.312111	0.0
7	0.689655	0.555556	0.439181	0.4
8	0.379310	0.777778	0.373325	0.0
9	0.034483	0.000000	0.431702	1.0
10	0.482759	0.555556	0.312111	0.8
11	0.379310	0.777778	0.023562	0.6
12	0.379310	0.777778	0.000000	0.6
13	0.689655	0.888889	0.270632	0.6
14	0.827586	0.777778	0.312111	0.8
15	0.379310	0.888889	0.128312	0.6
16	0.103448	0.333333	0.538098	0.2
17	0.000000	0.111111	0.416612	0.8
18	0.448276	0.444444	0.499427	0.8
19	0.655172	0.555556	0.312111	0.2
20	0.793103	0.888889	0.120628	0.6
21	0.413793	0.666667	0.472422	0.6
22	0.689655	0.777778	0.246516	0.0
23	0.068966	0.222222	0.312111	1.0
24	0.689655	0.777778	0.312111	0.0
25	0.551724	1.000000	0.224761	0.6
26	0.724138	0.888889	0.385133	0.6
27	0.655172	0.888889	0.312111	0.6
28	0.344828	0.777778	0.586144	0.6
29	0.379310	0.777778	0.068038	0.6
30	0.344828	0.666667	0.077082	0.0
31	0.379310	0.333333	0.562662	0.2
32	0.241379	0.555556	0.580750	0.8
33	1.000000	0.777778	0.312111	0.0
34	0.000000	0.222222	0.464907	0.2
35	0.793103	0.777778	0.325378	1.0

	0	1	2	3
36	0.344828	1.000000	0.131577	0.6
37	0.758621	0.888889	0.138993	0.6
38	1.000000	0.666667	0.379676	0.6
39	0.862069	0.777778	0.312111	0.0
40	0.137931	1.000000	0.216137	0.6
41	1.000000	0.888889	0.372448	0.6
42	0.206897	0.444444	0.746605	0.0
43	0.620690	0.333333	0.312111	0.2
44	0.931034	0.888889	0.071169	0.6
45	0.517241	1.000000	0.294302	0.6
46	0.586207	0.222222	0.312111	0.6
47	0.275862	0.333333	1.000000	0.6
48	0.448276	0.222222	0.183889	0.6
49	0.896552	1.000000	0.220601	0.6

Train Test Split

In [210…
```
from sklearn.model_selection import train_test_split
```

In [211…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_st
```

In [212…
```
X_train
```

Out[212]:

	0	1	2	3
12	11.0	2020.0	8506.000000	3.0
27	19.0	2021.0	43396.923077	3.0
33	29.0	2020.0	43396.923077	0.0
16	3.0	2016.0	68660.000000	1.0
2	28.0	2022.0	32879.000000	3.0
25	16.0	2022.0	33632.000000	3.0
14	24.0	2020.0	43396.923077	4.0
30	10.0	2019.0	17123.000000	0.0
5	11.0	2020.0	29026.000000	0.0
32	7.0	2018.0	73428.000000	4.0
1	5.0	2021.0	15138.000000	0.0
29	11.0	2020.0	16112.000000	3.0
35	23.0	2020.0	44880.000000	5.0
41	29.0	2021.0	50142.000000	3.0
19	19.0	2018.0	43396.923077	1.0
37	22.0	2021.0	24044.000000	3.0
10	14.0	2018.0	43396.923077	4.0
4	11.0	2020.0	27496.000000	3.0
6	9.0	2019.0	43396.923077	0.0
3	11.0	2020.0	37190.000000	3.0
20	23.0	2021.0	21991.000000	3.0
26	21.0	2021.0	51560.000000	3.0
38	29.0	2019.0	50950.000000	3.0
21	12.0	2019.0	61318.000000	3.0
42	6.0	2017.0	91969.000000	0.0
31	11.0	2016.0	71406.000000	1.0
34	0.0	2015.0	60478.000000	1.0
7	20.0	2018.0	57602.000000	2.0
49	26.0	2022.0	33167.000000	3.0
11	11.0	2020.0	11140.000000	3.0
18	13.0	2017.0	64337.000000	4.0
43	18.0	2016.0	43396.923077	1.0
22	20.0	2020.0	36064.000000	0.0
8	11.0	2020.0	50240.000000	0.0
45	15.0	2022.0	41406.000000	3.0
15	11.0	2021.0	22850.000000	3.0

	0	1	2	3
40	4.0	2022.0	32668.000000	3.0

In [213…
```
X_train.shape
```

Out[213]:
```
(37, 4)
```

In [214…
```
X_test
```

Out[214]:

	0	1	2	3
36	10.0	2022.0	23215.000000	3.0
47	8.0	2016.0	120296.000000	3.0
28	10.0	2020.0	74031.000000	3.0
9	1.0	2013.0	56766.000000	5.0
13	20.0	2021.0	38760.000000	3.0
0	15.0	2022.0	41406.000000	3.0
44	27.0	2021.0	16462.000000	3.0
46	17.0	2015.0	43396.923077	3.0
39	25.0	2020.0	43396.923077	0.0
23	2.0	2015.0	43396.923077	5.0
24	20.0	2020.0	43396.923077	0.0
48	13.0	2015.0	29063.000000	3.0
17	0.0	2014.0	55079.000000	4.0

In [215…
```
X_test.shape
```

Out[215]:
```
(13, 4)
```

In [216…
```
Y_train
```

Out[216]:

	0
12	21950.275
27	29990.000
33	20998.000
16	17998.000
2	24988.000
25	19988.000
14	39998.000
30	21950.275
5	21000.000
32	19998.000
1	38008.000
29	20998.000
35	21950.275
41	17998.000
19	18285.000
37	20998.000
10	30998.000
4	21950.275
6	20998.000
3	18998.000
20	21950.275
26	16488.000
38	20998.000
21	26998.000
42	14998.000
31	16498.000
34	16998.000
7	18887.000
49	20222.000
11	20998.000
18	19598.000
43	20994.000
22	26839.000
8	19498.000
45	15988.000
15	21998.000

	0
40	21950.275

In [217…
```
Y_train.shape
```

Out[217]:
```
(37, 1)
```

In [218…
```
Y_test
```

Out[218]:

	0
36	27798.000
47	21950.275
28	23998.000
9	17998.000
13	29998.000
0	15988.000
44	21950.275
46	15998.000
39	23998.000
23	21950.275
24	21950.275
48	23998.000
17	17998.000

In [219…
```
Y_test.shape
```

Out[219]:
```
(13, 1)
```

In [ ]:
```

```