

Capstone Project (Cleaning)

- Project : Real Estate price prediction

Data Cleaning

- We have 2 datasets:
 - Flats (~3000 rows)
 - Housing (~1000 rows)
- **Step 1:** Perform cleaning on both the datasets
- **Step 2:** Merge the 2 datasets
- **Step 3:** Perform cleaning on the merged dataset

Step 1: Perform cleaning on both the datasets

Cleaning Flats

- Removed incomplete rows

9	3 BHK Flat	https://w	Smart Wo	2.24 Crore	â, 14,400/sq.ft.	Super Bu	3 Bedroom	3 Bathroom	2 Balconie	Others	Si
0	2 BHK Flat	https://w	Smart Wo	1.57 Crore	â, 13,652/sq.ft.	Built Up ar	2 Bedroom	2 Bathroom	2 Balconie	Study Roo	Si
1	2 BHK Flat	https://w	Signature Global Solera	3.7 â	...						
2	4 BHK Flat	https://w	Tulip Monsella		â, 13,198/sq.ft.						
3	2 BHK Flat	https://w	My Home		â, 14,400/sq.ft.						
4	2 BHK Flat	https://w	Breez Global Hill Vie	â, 15,470/sq.ft.							
5	4 BHK Flat	https://w	Tulip Mon	7.4 Crore	â, 13,198/sq.ft.	Carpet are	4 Bedroom	4 Bathroom	3 Balconies		Si
6	2 BHK Flat	https://w	My Home	22 Lac	â, 14,400/sq.ft.	Carpet are	2 Bedroom	2 Bathroom	1 Balcony		N
7	2 BHK Flat	https://w	Breez Glo	22 Lac	â, 15,470/sq.ft.	Built Up ar	2 Bedroom	2 Bathroom	1 Balcony		1

Python Code

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

- This will display all rows and columns.

```
df.shape
```

```
(3008, 20)
```

```
# info
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3008 entries, 0 to 3007
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   property_name          3008 non-null   object
1   link                   3008 non-null   object
2   society                 3007 non-null   object
3   price                   3007 non-null   object
4   area                    2996 non-null   object
5   areaWithType            3008 non-null   object
6   bedRoom                 3008 non-null   object
7   bathroom                3008 non-null   object
8   balcony                 3008 non-null   object
9   additionalRoom          1694 non-null   object
10  address                  3002 non-null   object
11  floorNum                 3006 non-null   object
12  facing                   2127 non-null   object
13  agePossession            3007 non-null   object
14  nearbyLocations          2913 non-null   object
15  description              3008 non-null   object
16  furnishDetails           2203 non-null   object
17  features                 2594 non-null   object
18  rating                   2676 non-null   object
19  property_id              3008 non-null   object
dtypes: object(20)
memory usage: 470.1+ KB

```

Check for duplicates:

```
df.duplicated().sum()
```

Output: 0

Check for missing values:

```
df.isnull().sum()
```

```
property_name    0
link             0
society          1
price            1
area            12
areaWithType     0
bedRoom          0
bathroom         0
balcony          0
additionalRoom   1314
address          6
floorNum         2
facing           881
agePossession    1
nearbyLocations  95
description       0
furnishDetails   805
features         414
rating           332
property_id      0
dtype: int64
```

Remove Unnecessary Columns:

```
df.drop(columns=['link','property_id'], inplace=True)
```

Rename the **area** column to → **price_per_sqft**

```
df.rename(columns={'area':'price_per_sqft'},inplace=True)
```

Dealing with the '*Society*' Column:

```
df['society'].value_counts()
```

society	
SS The Leaf	3.8 ★ 73
Tulip Violet	4.3 ★ 40
Shapoorji Pallonji Joyville Gurugram	4.0 ★ 39
Signature Global Park	4.0 ★ 36
Shree Vardhman Victoria	3.8 ★ 35
Tulip Violet	4.2 ★ 33
Emaar MGF Emerald Floors Premier	3.8 ★ 32
Smart World Gems	32
Smart World Orchard	32
Paras Dews	31
DLF The Ultima	4.0 ★ 31
DLF Regal Gardens	3.9 ★ 30
Shree Vardhman Flora	3.8 ★ 29
M3M Woodshire	4.0 ★ 29

```
df['society'].value_counts().shape
```

- There are **636** societies in total.

Remove the ratings from name:

DLF The Ultima	4.0 ★
DLF Regal Gardens	3.9 ★
Shree Vardhman Flora	3.8 ★
M3M Woodshire	4.0 ★
La Vida by Tata Housing	
Signature Global Solera	3.7 ★
Godrej Nature Plus	
BPTP Terra	3.8 ★
Emaar Gurgaon Greens	4.1 ★
Vatika Gurgaon 21	3.7 ★
Experion The Heartsong	3.9 ★
Eldeco Accolade	3.8 ★
DLF New Town Heights	3.9 ★
Bestech Park View Residency	3.9 ★

```
import re
df['society'] = df['society'].apply(lambda name: re.sub(r'\d+(\.\d+)?\s?★', '', str(
```

\d	Matches any digit (equivalent to [0-9]).	r"\d"	"1", "2", "9", etc.
+	Matches 1 or more occurrences of the preceding pattern.	r"lo+" (<i>It can't have another character except "o" before the string ends.</i>)	"lo", "loo", "looo", etc. No match: l lol loO
?	Matches 0 or 1 occurrence of the preceding pattern.- Preceding character is optional .	r"lo?"r"colou?r"	"l", "lo" Doesn't Match: "loo", "looo""colour", "color"
\s	Matches any whitespace character (spaces, tabs, newlines).	r"\s"	" ", "\t", "\n", etc.

.apply(lambda name: ...) :

- `.apply()` is used to apply a function (in this case, a **lambda function**) to each element of the `'society'` column.
- The lambda function takes each value in the `'society'` column (each value is referred to as `name` inside the lambda function).

str(name) :

- The `str(name)` is converting the value of `name` into a string format. This ensures that even if the value is not a string (e.g., an integer, float, or NaN), it will be treated as a string for further processing.
- It ensures that `re.sub()` can work with it.

`re.sub(r'\d+(\.\d+)?\s?★', '', str(name)) :`

- `re.sub(pattern, replacement, string)` is used to substitute parts of the string that match a regular expression (`pattern`) with a specified `replacement` .
- `re.sub()` is a function from the `re` module that **substitutes (replaces)** occurrences of a pattern with a replacement string.

`r'\d+(\.\d+)?\s?★'`

- `\d+` : Matches one or more digits.
- `(\.\d+)?` : Matches an optional decimal part (a dot followed by one or more digits). The `?` makes it optional.
- `\s?` : Matches an optional whitespace character.
- `★` : Matches the star character.

`"` is the replacement string (an empty string), meaning the matched pattern will be removed.

`.strip()`

- After the `re.sub()` operation, `.strip()` is used to remove any leading or trailing whitespace that might remain in the string after the pattern is removed.

`.str.lower()`

- `.str.lower()` converts the entire string to lowercase.

```
df['society'].value_counts().shape
```

```
(602,)
```

- We reduced categories from 636 → 602

Price Column

```
df['price'].value_counts()
```

```
price
1.25 Crore    79
1.1 Crore     61
1.4 Crore     60
1.5 Crore     59
1.2 Crore     59
90 Lac        58
1.3 Crore     57
95 Lac        53
2 Crore       51
1.75 Crore    47
1 Crore       46
1.6 Crore     43
1.35 Crore    41
1.55 Crore    40
1.9 Crore     40
75 Lac        38
1.65 Crore    38
1.8 Crore     37
1.7 Crore     37
80 Lac        36
2.2 Crore     34
50 Lac        33
```

We have price on request values

```
25 Lac        11
1.08 Crore    11
Price on Request X 11
28 Lac        10
1.28 Crore    10
87 Lac        10
```

- We have to remove it because our model is price prediction model

```
df = df[df['price'] != 'Price on Request']
```


- This effectively filters out entries that have this placeholder text and keeps only the rows with valid price values.

Convert Lac → Crore(INT)

```
def treat_price(x):  
    if type(x) == float:  
        return x  
    else:  
        if x[1] == 'Lac':  
            return round(float(x[0])/100,2)  
        else:  
            return round(float(x[0]),2)
```

- `x[0]` : The numerical part of the price.
- `x[1]` : The unit of the price (either 'Lac' or 'Cr').

'Lac' Handling:

- `if x[1] == 'Lac':` checks if the unit is 'Lac' (Lakhs).
- `return round(float(x[0])/100, 2) :`
 - `float(x[0])` converts the numerical part of the price to a float.
 - `/ 100` converts Lakhs to Crores (1 Crore = 100 Lakhs).
 - `round(..., 2)` rounds the result to two decimal places.

'Cr' Handling (or other):

- `else:` If the unit is not 'Lac' (it's assumed to be 'Cr' or some other unit representing Crores directly).
- `return round(float(x[0]), 2) :`
 - `float(x[0])` converts the numerical part to a float.
 - `round(..., 2)` rounds the result to two decimal places

 **BUT THE PRICE IS IN STRING FORMAT.**

- We have to `split` it first.

```
df['price'] = df['price'].str.split(' ').apply(treat_price)
```

Splitting the String (`x.split()`):

- This splits the string `x` by **spaces** into a list of parts. For example:
 - `'5 Lac'` becomes `['5', 'Lac']`.
 - `'5000 INR'` becomes `['5000', 'INR']`.
 - `'15.5'` becomes `['15.5']`.

```
df.head(5)
```

	property_name	society	price	price_per_sqft	areaWithType
0	2 BHK Flat in Krishna Colony	maa bhagwati residency	0.45	₹ 5,000/sq.ft.	Carpet area: 900 (83.61 sq.m.)
1	2 BHK Flat in Ashok Vihar	apna enclave	0.50	₹ 7,692/sq.ft.	Carpet area: 650 (60.39 sq.m.)
2	2 BHK Flat in Sohna	tulsiani easy in homes	0.40	₹ 6,722/sq.ft.	Carpet area: 595 (55.28 sq.m.)

price_per_sqft Column

```
# price_per_sqft
```

```
df['price_per_sqft'].value_counts()
```

```
price_per_sqft
₹ 10,000/sq.ft.    19
₹ 8,000/sq.ft.     16
₹ 12,500/sq.ft.    16
₹ 6,666/sq.ft.     13
₹ 5,000/sq.ft.     13
₹ 7,500/sq.ft.     12
₹ 8,333/sq.ft.     12
₹ 6,000/sq.ft.     11
₹ 8,461/sq.ft.      9
₹ 12,000/sq.ft.     8
₹ 7,000/sq.ft.      8
₹ 9,000/sq.ft.      7
₹ 11,111/sq.ft.     6
₹ 5,500/sq.ft.      6
₹ 6,578/sq.ft.      6
₹ 8,928/sq.ft.      6
```

- Clean the above data:

- Split with / → `df['price_per_sqft'].str.split('/')`
- Extract index [0] element (ex. ₹ 10,000) → `.str.get(0)`
- Replace the ₹ symbol with nothing → `.str.replace('₹','')`
- Remove comma in similar manner → `.str.replace(',','')`
- Remove white spaces → `.str.strip()`
- Convert to float → `.astype('float')`

```
df['price_per_sqft'] = df['price_per_sqft'].str.split('/').str.get(0).str.replace('₹','').s
```

property_name	society	price	price_per_sqft	ar
2 BHK Flat in Krishna Colony	maa bhagwati residency	0.45	5000.0	
2 BHK Flat in Ashok Vihar	apna enclave	0.50	7692.0	
2 BHK Flat in Sohna	tulsiani easy in homes	0.40	6722.0	

- We did not convert it into INT because there are missing values and it would show error.

Bedroom, bathroom & balcony columns

```
# bedrooms
df['bedRoom'].value_counts()
```

```
3 Bedrooms    1437
2 Bedrooms    944
4 Bedrooms    478
1 Bedroom     104
5 Bedrooms     31
6 Bedrooms      3
Name: bedRoom, dtype: int64
```

```
df[df['bedRoom'].isnull()]
```

	property_name	society	price	price_per_sqft	areaWithType	bedRoom	bathroom	balcony	additionalRoom	address	floorNum	facing	agePossession	nearbyLocations	c
849	2 BHK Flat in Sector 107 Gurgaon	signature global solera	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
850	4 BHK Flat in Sector 53 Gurgaon	tulip monsell	NaN	33198.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
851	2 BHK Flat in New Palam Vihar	my home	NaN	4400.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
852	2 BHK Flat in Sohna	breez global hill view	NaN	5470.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
922	3 BHK Flat in Sector 99A Gurgaon	pareena coban residences	NaN	5759.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
923	1 BHK Flat in Golf Course Ext Road	ikon tower baani city centre	NaN	12437.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
	2 BHK Flat in														

- We can remove these rows as they dont have any informatoin.

```
df = df[~df['bedRoom'].isnull()]
```

~ → is not

- Remove the word **bedroom** and convert into INT
 - eX. Convert `1 bedroom` → `1`

```
df['bedRoom'] = df['bedRoom'].str.split(' ').str.get(0).astype('int')
```

- Do the same on bathrooms & balcony

```
df['bathroom'].value_counts()
```

```
bathroom
2 Bathrooms    1044
3 Bathrooms    989
4 Bathrooms    636
5 Bathrooms    169
1 Bathroom     112
6 Bathrooms     42
7 Bathrooms      5
Name: count, dtype: int64
```

```
df['bathroom'].isnull().sum()
```

Output: 0

- No missing values

```
df['bathroom'] = df['bathroom'].str.split(' ').str.get(0).astype('int')
```

```
df['balcony'].value_counts()
```

```
balcony
3 Balconies    974
3+ Balconies   862
2 Balconies    749
1 Balcony      315
No Balcony      97
Name: count, dtype: int64
```

```
df['balcony'].isnull().sum()
```

Output: 0

- No missing values

```
df['balcony'] = df['balcony'].str.split(' ').str.get(0).str.replace('No','0')
```

Note: We did not convert this into INT because there's a 3+ value

Additional Room Column:

```
# additionalRoom  
df['additionalRoom'].value_counts()
```

```
additionalRoom  
Servant Room      629  
Study Room        232  
Others            179  
Pooja Room        132  
Study Room,Servant Room    81  
Store Room        76  
Pooja Room,Servant Room    60  
Servant Room,Others    52  
Servant Room,Pooja Room    30  
Study Room,Others    27
```

```
df['additionalRoom'].value_counts().shape
```

Output: (49,)

```
(49,)
```

```
df['additionalRoom'].isnull().sum()
```

Output: 1305

- Replace NA → not available

```
df['additionalRoom'].fillna('not available',inplace=True)
df['additionalRoom'] = df['additionalRoom'].str.lower()
```

floorNum:

```
df['floorNum']
```

```
4th  of 4 Floors
1st  of 3 Floors
12nd of 14 Floors
2nd  of 4 Floors
5th  of 8 Floors
3rd  of 3 Floors
5th  of 25 Floors
14th of 27 Floors
2nd  of 3 Floors
31st of 40 Floors
1st  of 4 Floors
4th  of 13 Floors
4th  of 15 Floors
3rd  of 20 Floors
```

```
df['floorNum'] = df['floorNum'].str.split(' ').str.get(0).replace('Ground','0').str.rep
```

- `df['floorNum'].str.split(' ')` :
 - Splits each value in the `'floorNum'` column by spaces.
- `.str.get(0)` :
 - Selects the first part of the split result (i.e., the first word).
- `.replace('Ground','0')` :
 - Replaces the word `'Ground'` with `'0'`.
- `.str.replace('Basement','-1')` :
 - Replaces the word `'Basement'` with `'-1'`.
- `.str.replace('Lower','0')` :

- Replaces the word 'Lower' with '0'.
- `.str.extract(r'(\d+)')` :
 - Extracts the first sequence of digits from the modified string (this converts the result to a number).

facing Column:

```
df['facing'].value_counts()
```

```
facing
North-East    505
East          490
North         301
South         203
West          183
North-West    162
South-East    144
South-West    135
Name: count, dtype: int64
```

```
df['facing'].isnull().sum()
```

Output: 874

- Replace missing values with NA

```
df['facing'].fillna('NA',inplace=True)
```

Area:

- **Calculate area of the flat:**
- We'll divide `price` with `price_per_sqft`
 - `price / price_per_sqft = area`

```
df.insert(loc=4,column='area',value=round((df['price']*10000000)/df['price_per_
```

- We inserted this column at loc 4

society	price	price_per_sqft	area	areaWithType	b
maahatwadi bhagwati residency	0.45	5000.0	900.0	Carpet area: 900 (83.61 sq.m.)	
apna enclave	0.50	7692.0	650.0	Carpet area: 650 (60.39 sq.m.)	
tulsiani easy in homes	0.40	6722.0	595.0	Carpet area: 595 (55.28 sq.m.)	

Export to csv:

```
df.to_csv('flats_cleaned.csv',index=False)
```



Did the same thing with **houses** dataset.

Export house data to csv:

```
df.to_csv('house_cleaned.csv',index=False)
```

Step 2: Merge the 2 datasets:

```
flats = pd.read_csv('flats_cleaned.csv')  
houses = pd.read_csv('house_cleaned.csv')
```

```
df = pd.concat([flats,houses],ignore_index=True)
```

Shuffle the data

```
df = df.sample(df.shape[0],ignore_index=True)
```

- `df.sample()` : This is a Pandas DataFrame method that returns a random sample of rows from the DataFrame `df` .
- `df.shape[0]` : This gets the number of rows in the DataFrame `df` .
- Therefore, `df.sample(df.shape[0], ...)` is instructing Pandas to select a random sample that includes *all* rows of the DataFrame.

```
df.head()
```

	property_name	property_type	society	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	additionalRoom	address	floorNum	facing	agePosses
0	4 Bedroom House for sale in Sector 70A Gurgaon	house	bptp visionnaire	5.40	20690.0	2610.0	Plot area 290(242.48 sq.m.)	4 Bedrooms	5 Bathrooms	3+	pooja room,store room,study room,servant room	290 Sqyd Duplex Villa, Sector 70A Gurgaon, Gur...	3.0	North	1 to 5 Yea
1	3 BHK Flat in Sector 92 Gurgaon	flat	sare homes	0.58	4098.0	1415.0	Super Built up area 1415(131.46 sq.m.)	3	3	3	not available	E020304, Sector 92 Gurgaon, Gurgaon, Haryana	3.0	North-East	0 to 1 Yea
2	3 Bedroom House for sale in Hira Nagar	house	independent	1.05	9722.0	1080.0	Built Up area: 1080 (100.34 sq.m.)	3 Bedrooms	2 Bathrooms	0	not available	Hira Nagar, Gurgaon, Haryana	1.0	NaN	unde
3	4 BHK Flat in Sector 61 Gurgaon	flat	pioneer park	3.60	12627.0	2851.0	Super Built up area 2851(264.87 sq.m.)	4	4	3	servant room	Sector 61 Gurgaon, Gurgaon, Haryana	3.0	North	5 to 10

Export this to csv:

```
df.to_csv('gurgaon_properties.csv',index=False)
```

- Now, we'll clean this data

Step 3: Clean the merged data

```
df = pd.read_csv('gurgaon_properties.csv')
```

```
df.shape
```

```
Output:(3961, 20)
```

```
df.isnull().sum()
```

```
property_name      0
property_type      0
society            1
price              20
price_per_sqft     20
area               20
areaWithType       0
bedRoom            0
bathroom           0
balcony            0
additionalRoom     0
address            11
floorNum           21
facing             1177
agePossession      1
nearbyLocations    207
description         0
furnishDetails     1032
features           709
rating             450
dtype: int64
```

Add additional column → **sector**

```
df.insert(loc=3,column='sector',value=df['property_name'].str.split('in').str.get(1).str.replace('Gurgaon','').str.strip())
```

	property_name	property_type	society	sector	price	price_per_sqft
0	3 BHK Flat in Sector 65 Gurgaon	flat	m3m heights	Sector 65	2.86	14000.0
1	5 Bedroom House for sale in Sector 66 Gurgaon	house	emaar mgf marbella	Sector 66	19.00	31666.0
2	3 BHK Flat in Sector 37D Gurgaon	flat	ramprastha primera	Sector 37D	1.08	6000.0

Clean the sector column:

```
df['sector'].value_counts()
```

```

sector
sohna 163
sector 102 113
sector 85 110
sector 92 104
sector 69 94
sector 81 90
sector 90 90
sector 65 90
sector 109 88
sector 79 80
sector 83 69
sector 37d 68
sector 86 67
sector 104 66
sector 107 60
sector 108 59
sector 95 57
sector 56 57
sector 48 56
sector 89 56
sector 70a 54
nirvana country 53
sector 70 53
sector 37c 53
...
rajiv colony 1
naharpur rupa 1
near khandsa road 1
ashok vihar phase 3 1
Name: count, dtype: int64
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output

```

Replace the society name with sector:

```

df['sector'] = df['sector'].str.replace('dharam colony','sector 12')
df['sector'] = df['sector'].str.replace('krishna colony','sector 7')
df['sector'] = df['sector'].str.replace('suncity','sector 54')
df['sector'] = df['sector'].str.replace('prem nagar','sector 13')
df['sector'] = df['sector'].str.replace('mg road','sector 28')

```

```
df['sector'] = df['sector'].str.replace('gandhi nagar','sector 28')  
df['sector'] = df['sector'].str.replace('laxmi garden','sector 11')  
df['sector'] = df['sector'].str.replace('shakti nagar','sector 11')
```

Drop unnecessary columns

- property_name, address, description, rating

```
df.drop(columns=['property_name', 'address', 'description', 'rating'],inplace=True)
```

Extract to CSV:

```
df.to_csv('gurgaon_properties_cleaned_v1.csv',index=False)
```