In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

unzip the file

In [63]:
```
from zipfile import ZipFile

with ZipFile('wine.zip', 'r') as zip_object:
    zip_object.extractall()


print(zip_object.namelist())
```

```
['Index', 'wine.data', 'wine.names']
```

after unzip convert the data into a dataframe by counting the columns

In [57]:
```
df = pd.read_csv("wine.data", header=None)
```

In [58]:
```
df
```

Out[58]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93
...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60

178 rows × 14 columns

In [59]:
```
df.columns = [
    'Class',
    'Alcohol',
    'Malic_Acid',
    'Ash',
    'Alcalinity_of_Ash',
    'Magnesium',
    'Total_Phenols',
    'Flavanoids',
    'Nonflavanoid_Phenols',
    'Proanthocyanins',
    'Color_Intensity',
    'Hue',
    'OD280/OD315_of_Diluted_Wines',
    'Proline'
]
```

In [60]:

```
df.head(10)
```

Out[60]:

	Class	Alcohol	Malic_Acid	Ash	Alcalinity_of_Ash	Magnesium	Total_Pheno
0	1	14.23	1.71	2.43	15.6	127	2.8
1	1	13.20	1.78	2.14	11.2	100	2.6
2	1	13.16	2.36	2.67	18.6	101	2.8
3	1	14.37	1.95	2.50	16.8	113	3.8
4	1	13.24	2.59	2.87	21.0	118	2.8
5	1	14.20	1.76	2.45	15.2	112	3.2
6	1	14.39	1.87	2.45	14.6	96	2.5
7	1	14.06	2.15	2.61	17.6	121	2.6
8	1	14.83	1.64	2.17	14.0	97	2.8
9	1	13.86	1.35	2.27	16.0	98	2.9

In [54]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Class                         178 non-null    int64  
 1   Alcohol                       178 non-null    float64
 2   Malic_Acid                    178 non-null    float64
 3   Ash                           178 non-null    float64
 4   Alcalinity_of_Ash             178 non-null    float64
 5   Magnesium                     178 non-null    int64  
 6   Total_Phenols                 178 non-null    float64
 7   Flavanoids                    178 non-null    float64
 8   Nonflavanoid_Phenols          178 non-null    float64
 9   Proanthocyanins               178 non-null    float64
 10  Color_Intensity               178 non-null    float64
 11  Hue                           178 non-null    float64
 12  OD280/OD315_of_Diluted_Wines  178 non-null    float64
 13  Proline                       178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

In [61]:

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

Out[61]:

```
Class                           0
Alcohol                         0
Malic_Acid                      0
Ash                             0
Alcalinity_of_Ash               0
Magnesium                       0
Total_Phenols                   0
Flavanoids                      0
Nonflavanoid_Phenols            0
Proanthocyanins                 0
Color_Intensity                 0
Hue                             0
OD280/OD315_of_Diluted_Wines    0
Proline                         0
dtype: int64
```

In [ ]:

In [ ]:

In [64]:

```
df1= pd.read_csv('miss.csv')
df1
```

Out[64]:

	area_type	availability	location	size	society	total_sqft
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200
...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550

13320 rows × 9 columns

In [65]:

```
df1.head()
```

Out[65]:

	area_type	availability	location	size	society	total_sqft	bat
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.

In [66]:

```
df1.describe  #find descriptive status of data
```

Out[66]:

```
<bound method NDFrame.describe of                    area_type   availabilit
y                   location  \
0      Super built-up  Area         19-Dec  Electronic City Phase II
1                Plot  Area  Ready To Move          Chikka Tirupathi
2            Built-up  Area  Ready To Move               Uttarahalli
3      Super built-up  Area  Ready To Move        Lingadheeranahalli
4      Super built-up  Area  Ready To Move                  Kothanur
...                     ...           ...                       ...
13315        Built-up  Area  Ready To Move                Whitefield
13316  Super built-up  Area  Ready To Move             Richards Town
13317        Built-up  Area  Ready To Move     Raja Rajeshwari Nagar
13318  Super built-up  Area         18-Jun           Padmanabhanagar
13319  Super built-up  Area  Ready To Move              Doddathoguru

            size  society total_sqft  bath  balcony   price
0          2 BHK  Coomee        1056   2.0      1.0   39.07
1      4 Bedroom  Theanmp       2600   5.0      3.0  120.00
2          3 BHK      NaN       1440   2.0      3.0   62.00
3          3 BHK  Soiewre       1521   3.0      1.0   95.00
4          2 BHK      NaN       1200   2.0      1.0   51.00
...          ...      ...        ...   ...      ...     ...
13315  5 Bedroom  ArsiaEx       3453   4.0      0.0  231.00
13316      4 BHK      NaN       3600   5.0      NaN  400.00
13317      2 BHK  Mahla T       1141   2.0      1.0   60.00
13318      4 BHK  SollyCl       4689   4.0      1.0  488.00
13319      1 BHK      NaN        550   1.0      1.0   17.00

[13320 rows x 9 columns]>
```

In [67]:

```
df1.describe()
```

Out[67]:

	bath	balcony	price
count	13247.000000	12711.000000	13320.000000
mean	2.692610	1.584376	112.565627
std	1.341458	0.817263	148.971674
min	1.000000	0.000000	8.000000
25%	2.000000	1.000000	50.000000
50%	2.000000	2.000000	72.000000
75%	3.000000	2.000000	120.000000
max	40.000000	3.000000	3600.000000

In [68]:

```
df1.describe().sum()
```

Out[68]:

```
bath       13299.034068
balcony    12721.401639
price      17431.537300
dtype: float64
```

In [69]:

```
df1.info()  #summary of data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

In [70]:

```
df1.count
```

Out[70]:

```
<bound method DataFrame.count of                    area_type   availability
location  \
0      Super built-up  Area         19-Dec  Electronic City Phase II
1                Plot  Area  Ready To Move          Chikka Tirupathi
2            Built-up  Area  Ready To Move               Uttarahalli
3      Super built-up  Area  Ready To Move        Lingadheeranahalli
4      Super built-up  Area  Ready To Move                  Kothanur
...                   ...            ...                       ...
13315        Built-up  Area  Ready To Move                Whitefield
13316  Super built-up  Area  Ready To Move             Richards Town
13317        Built-up  Area  Ready To Move     Raja Rajeshwari Nagar
13318  Super built-up  Area         18-Jun           Padmanabhanagar
13319  Super built-up  Area  Ready To Move              Doddathoguru

            size  society total_sqft  bath  balcony   price
0          2 BHK  Coomee        1056   2.0      1.0   39.07
1      4 Bedroom  Theanmp       2600   5.0      3.0  120.00
2          3 BHK      NaN       1440   2.0      3.0   62.00
3          3 BHK  Soiewre       1521   3.0      1.0   95.00
4          2 BHK      NaN       1200   2.0      1.0   51.00
...          ...      ...        ...   ...      ...     ...
13315  5 Bedroom  ArsiaEx       3453   4.0      0.0  231.00
13316      4 BHK      NaN       3600   5.0      NaN  400.00
13317      2 BHK  Mahla T       1141   2.0      1.0   60.00
13318      4 BHK  SollyCl       4689   4.0      1.0  488.00
13319      1 BHK      NaN        550   1.0      1.0   17.00

[13320 rows x 9 columns]>
```

In [71]:

```
df1.dtypes
```

Out[71]:

```
area_type        object
availability     object
location         object
size             object
society          object
total_sqft       object
bath            float64
balcony         float64
price           float64
dtype: object
```

In [73]:

```
df1.shape
```

Out[73]:

```
(13320, 9)
```

In [74]:

```
df1.columns
```

Out[74]:

```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

In [75]:

```
df1.isna().sum()
```

```
Out[75]: area_type          0
         availability       0
         location           1
         size              16
         society         5502
         total_sqft         0
         bath              73
         balcony          609
         price              0
         dtype: int64
```

```
In [76]: df1.isnull().sum()
```

```
Out[76]: area_type          0
         availability       0
         location           1
         size              16
         society         5502
         total_sqft         0
         bath              73
         balcony          609
         price              0
         dtype: int64
```

```
In [79]: df1.isnull().sum().sum()
```

```
Out[79]: np.int64(6201)
```

percentage of missing value for each column

```
In [80]: (df1.isnull().sum()*100/df.index.size).round(2).sort_values(ascending=False)
```

```
Out[80]: society         3091.01
         balcony          342.13
         bath              41.01
         size               8.99
         location           0.56
         area_type          0.00
         availability       0.00
         total_sqft         0.00
         price              0.00
         dtype: float64
```

data missing value chceck using mean approach

```
In [81]: df1.isnull().mean()*100
```

Out[81]:

```
area_type        0.000000
availability     0.000000
location         0.007508
size             0.120120
society         41.306306
total_sqft       0.000000
bath             0.548048
balcony          4.572072
price            0.000000
dtype: float64
```

total no. of rows with null values

In [90]:

```
df1.isnull().sum(axis=1).sum()
```

Out[90]:

```
np.int64(6201)
```

In [93]:

```
cols= [var for var in df1.columns if df1[var].isnull().mean()< 0.05 and df1[
cols
```

Out[93]:

```
['location', 'size', 'bath', 'balcony']
```

Include the column only if: It has less than 5% missing values, and it has more than 0% missing values (i.e., at least one missing value).

find index of missing values in dataframe

In [91]:

```
df1[df1.isnull().any(axis=1)].index
```

Out[91]:

```
Index([    2,     4,     6,     7,     8,     9,    10,    13,    19,    2
0,
       ...
       13303, 13305, 13306, 13307, 13309, 13310, 13311, 13312, 13316, 1331
9],
      dtype='int64', length=5824)
```

find index of missing values in any one column

In [96]:

```
null_socity= df1.loc[df1['society'].isnull()].index
```

```
null_socity
```

Out[96]:

```
Index([    2,     4,     8,     9,    10,    13,    19,    20,    23,    2
5,
       ...
       13302, 13303, 13305, 13306, 13307, 13310, 13311, 13312, 13316, 1331
9],
      dtype='int64', length=5502)
```

In [97]:

```
null_balcony= df1.loc[df1['balcony'].isnull()].index
null_balcony
```

Out[97]:

```
Index([    6,     7,     9,    34,    40,    45,    56,    81,   140,   14
6,
       ...
       13213, 13217, 13232, 13240, 13247, 13277, 13279, 13306, 13309, 1331
6],
      dtype='int64', length=609)
```

In [98]:

```
count1= df1['balcony'].isnull().sum()
count1
```

Out[98]:

```
np.int64(609)
```

In [99]:

```
count2= df1['society'].isnull().sum()
count2
```

Out[99]:

```
np.int64(5502)
```

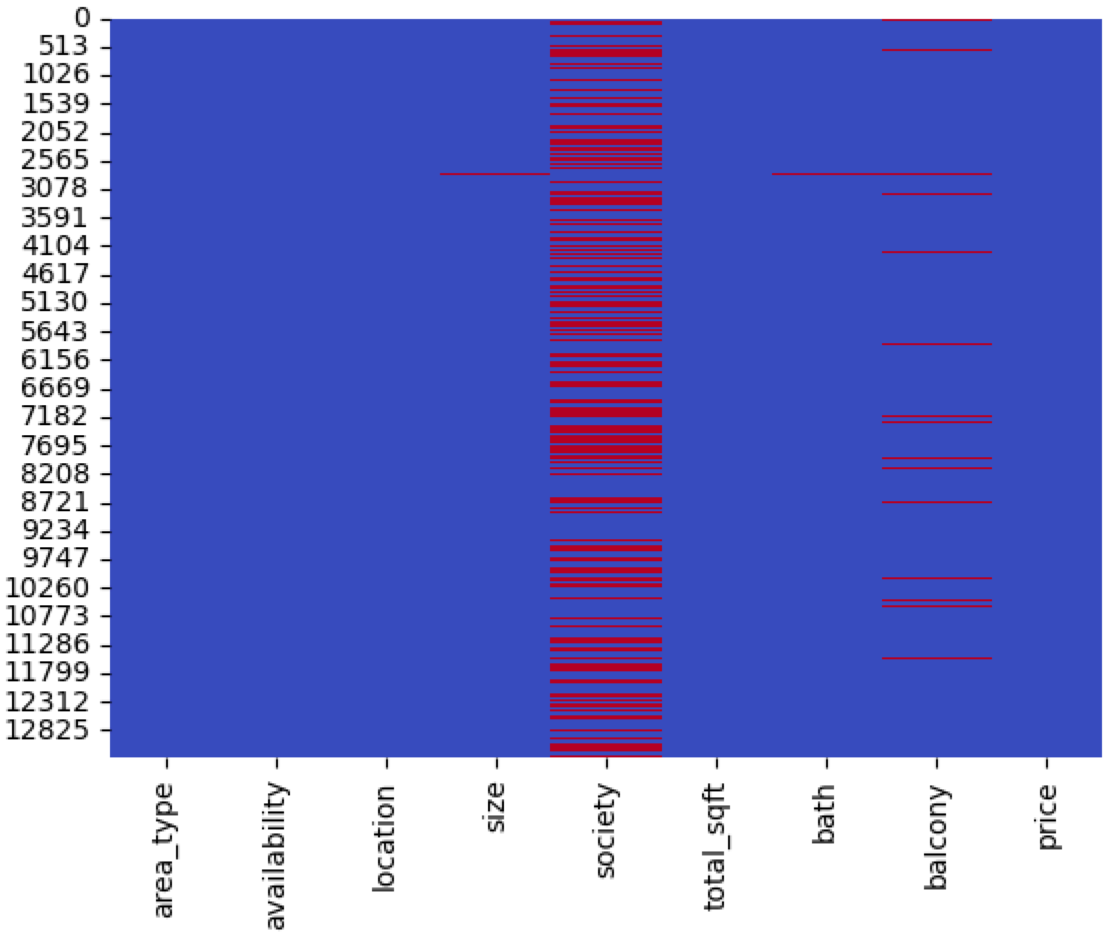
checking the data with their missing values in visual mode

In [104…

```
sns.heatmap(df1.isnull(), cmap='coolwarm',cbar=False)
```

Out[104…

```
<Axes: >
```



“coolwarm” is a gradient from blue (cool) to red (warm) — it helps visually distinguish between missing and non-missing data.

method 1

sklearn library module imputer

In [105…

```
from sklearn.impute import SimpleImputer
```

In [139…

```
dfamp= df1.copy()
# Create an instance of a simple imputer with a specified strategy
imputer= SimpleImputer(strategy='mean')

imputer.fit(dfamp)

#transform the dataset to fill the missing values
```

```
x= imputer.transform(dfamp)


# convert the transformed data into a dataframe
df_imputed= pd.DataFrame(x, columns= dfamp.columns)
specific_rows= df_imputed.iloc[rows_to_display]
specific_rows[['bath', 'balcony']]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[139], line 5
      2 # Create an instance of a simple imputer with a specified strategy
      3 imputer= SimpleImputer(strategy='mean')
----> 5 imputer.fit(dfamp)
      7 #transform the dataset to fill the missing values
      8 x= imputer.transform(dfamp)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:1473, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1466     estimator._validate_params()
   1468 with config_context(
   1469     skip_parameter_validation=(
   1470         prefer_skip_nested_validation or global_skip_validation
   1471     )
   1472 ):
-> 1473     return fit_method(estimator, *args, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\impute\_base.py:421, in SimpleImputer.fit(self, X, y)
    403 @_fit_context(prefer_skip_nested_validation=True)
    404 def fit(self, X, y=None):
    405     """Fit the imputer on `X`.
    406 
    407     Parameters
   (...)
    419         Fitted estimator.
    420     """
--> 421     X = self._validate_input(X, in_fit=True)
    423     # default fill_value is 0 for numerical input and "missing_value"
    424     # otherwise
    425     if self.fill_value is None:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\impute\_base.py:348, in SimpleImputer._validate_input(self, X, in_fit)
    342 if "could not convert" in str(ve):
    343     new_ve = ValueError(
    344         "Cannot use {} strategy with non-numeric data:\n{}".format(
    345             self.strategy, ve
    346         )
    347     )
--> 348     raise new_ve from None
    349 else:
    350     raise ve

ValueError: Cannot use mean strategy with non-numeric data:
could not convert string to float: 'Super built-up  Area'
```

In [143…

```python
# Copy the original DataFrame
dfamp = df1.copy()

# Identify numeric columns
numeric_cols = dfamp.select_dtypes(include=['number']).columns
```

```
# Create an instance of a simple imputer with a specified strategy
imputer = SimpleImputer(strategy='mean')

# Fit and transform only the numeric columns
imputer.fit(dfamp[numeric_cols])
x_numeric = imputer.transform(dfamp[numeric_cols])

# Convert the transformed numeric data into a DataFrame
df_numeric_imputed = pd.DataFrame(x_numeric, columns=numeric_cols)

# Combine the imputed numeric data with the original non-numeric columns
df_imputed = pd.concat([df_numeric_imputed, dfamp.drop(columns=numeric_cols)

# Access specific rows and columns
rows_to_display = [0, 1, 2, 3, 4]
specific_rows = df_imputed.iloc[rows_to_display]
result = specific_rows[['society', 'size']]
result
```

Out[143…

	society	size
0	Coomee	2 BHK
1	Theanmp	4 Bedroom
2	NaN	3 BHK
3	Soiewre	3 BHK
4	NaN	2 BHK

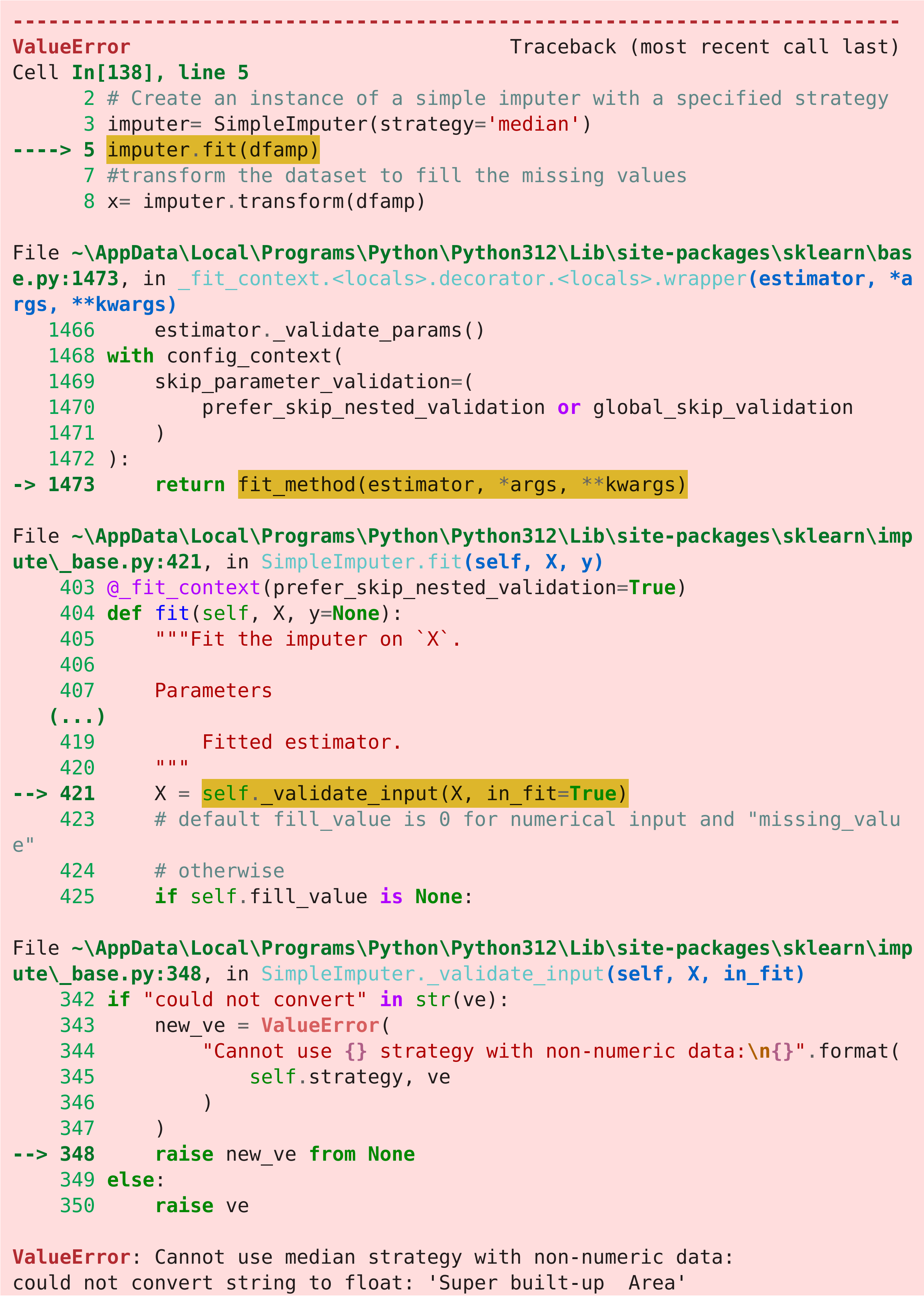
In [138…

```
dfamp= df1.copy()
# Create an instance of a simple imputer with a specified strategy
imputer= SimpleImputer(strategy='median')

imputer.fit(dfamp)

#transform the dataset to fill the missing values
x= imputer.transform(dfamp)


# convert the transformed data into a dataframe
df_imputed= pd.DataFrame(x, columns= dfamp.columns)
specific_rows= df_imputed.iloc[rows_to_display]
specific_rows[['bath', 'balcony']]
```



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[138], line 5
      2 # Create an instance of a simple imputer with a specified strategy
      3 imputer= SimpleImputer(strategy='median')
----> 5 imputer.fit(dfamp)
      7 #transform the dataset to fill the missing values
      8 x= imputer.transform(dfamp)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\bas
e.py:1473, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *a
rgs, **kwargs)
   1466     estimator._validate_params()
   1468 with config_context(
   1469     skip_parameter_validation=(
   1470         prefer_skip_nested_validation or global_skip_validation
   1471     )
   1472 ):
-> 1473     return fit_method(estimator, *args, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\imp
ute\_base.py:421, in SimpleImputer.fit(self, X, y)
    403 @_fit_context(prefer_skip_nested_validation=True)
    404 def fit(self, X, y=None):
    405     """Fit the imputer on `X`.
    406 
    407     Parameters
   (...)
    419         Fitted estimator.
    420     """
--> 421     X = self._validate_input(X, in_fit=True)
    423     # default fill_value is 0 for numerical input and "missing_valu
e"
    424     # otherwise
    425     if self.fill_value is None:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\imp
ute\_base.py:348, in SimpleImputer._validate_input(self, X, in_fit)
    342 if "could not convert" in str(ve):
    343     new_ve = ValueError(
    344         "Cannot use {} strategy with non-numeric data:\n{}".format(
    345             self.strategy, ve
    346         )
    347     )
--> 348     raise new_ve from None
    349 else:
    350     raise ve

ValueError: Cannot use median strategy with non-numeric data:
could not convert string to float: 'Super built-up  Area'
```

so this is only used for numeric data,
so that's why it shows this output

In [ ]:

```

```

In [131…

```python
dfono= df1.copy()
# Create an instance of a simple imputer with a specified strategy
imputer= SimpleImputer(strategy='constant', fill_value=0)

imputer.fit(dfono)

#transform the dataset to fill the missing values
y= imputer.transform(dfono)


# convert the transformed data into a dataframe
df4= pd.DataFrame(y, columns= dfono.columns)

row_to_display = 0
specific_rows= df4.iloc[row_to_display]
specific_rows[['society', 'size']]
```

Out[131…

```
society    Coomee
size        2 BHK
Name: 0, dtype: object
```

knn imputer:

In [126…

```python
from sklearn.impute import KNNImputer
```

In [127…

```python
dfknn= df1.copy()
dfknn
```

Out[127…

	area_type	availability	location	size	society	total_sqft
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200
...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550

13320 rows × 9 columns

In [130…

```
categorical_cols = dfknn.select_dtypes(include=['object']).columns
categorical_cols
```

Out[130…

```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft'],
      dtype='object')
```

In [136…

```
numeric_cols = dfamp.select_dtypes(include=['number']).columns
numeric_cols
```

Out[136…

```
Index(['bath', 'balcony', 'price'], dtype='object')
```

In [134…

```
imputer= KNNImputer(n_neighbors=5)
imputer.fit(dfknn)
#df_imputed= imputer.transform(dfknn)

imputed_values= imputer.transform(dfknn)
dfknn_imputed=pd.DataFrame(imputed_values,columns=dfknn.columns)
```

```
specific_rows= dfknn_imputed.iloc[rows_to_display]
specific_rows[['society', 'size']]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2116\1052786173.py in ?()
      1 imputer= KNNImputer(n_neighbors=5)
----> 2 imputer.fit(dfknn)
      3 #df_imputed= imputer.transform(dfknn)
      4 
      5 imputed_values= imputer.transform(dfknn)

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py
in ?(estimator, *args, **kwargs)
   1469                 skip_parameter_validation=(
   1470                     prefer_skip_nested_validation or global_skip_val
idation
   1471                 )
   1472             ):
-> 1473                 return fit_method(estimator, *args, **kwargs)

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\impute\_
knn.py in ?(self, X, y)
    226             force_all_finite = True
    227         else:
    228             force_all_finite = "allow-nan"
    229 
--> 230         X = self._validate_data(
    231             X,
    232             accept_sparse=False,
    233             dtype=FLOAT_DTYPES,

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py
in ?(self, X, y, reset, validate_separately, cast_to_ndarray, **check_param
s)
    629                 out = y
    630             else:
    631                 out = X, y
    632         elif not no_val_X and no_val_y:
--> 633             out = check_array(X, input_name="X", **check_params)
    634         elif no_val_X and not no_val_y:
    635             out = _check_y(y, **check_params)
    636         else:

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\va
lidation.py in ?(array, accept_sparse, accept_large_sparse, dtype, order, co
py, force_writeable, force_all_finite, ensure_2d, allow_nd, ensure_min_sampl
es, ensure_min_features, estimator, input_name)
   1009                         )
   1010                     array = xp.astype(array, dtype, copy=False)
   1011                 else:
   1012                     array = _asarray_with_order(array, order=order,
dtype=dtype, xp=xp)
-> 1013             except ComplexWarning as complex_warning:
   1014                 raise ValueError(
   1015                     "Complex data not supported\n{}\n".format(array)
   1016                 ) from complex_warning

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\_a
```

```
rray_api.py in ?(array, dtype, order, copy, xp, device)
    747         # Use NumPy API to support order
    748         if copy is True:
    749             array = numpy.array(array, order=order, dtype=dtype)
    750         else:
--> 751             array = numpy.asarray(array, order=order, dtype=dtype)
    752 
    753         # At this point array is a NumPy ndarray. We convert it to a
n array
    754         # container that is consistent with the input's namespace.

~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\gene
ric.py in ?(self, dtype, copy)
   2149     def __array__(
   2150         self, dtype: npt.DTypeLike | None = None, copy: bool_t | Non
e = None
   2151     ) -> np.ndarray:
   2152         values = self._values
-> 2153         arr = np.asarray(values, dtype=dtype)
   2154         if (
   2155             astype_is_view(values.dtype, arr.dtype)
   2156             and using_copy_on_write()

ValueError: could not convert string to float: 'Super built-up  Area'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: