Data Cleaning

Import Python Library

We will use a library: Pandas and numpy.

```
#Importing libraries
import pandas as pd
import numpy as np
```

Reading a .csv file.

After importing the libraries we read the csv file into a Pandas dataframe.

Let's consider the following data present in the file named **property.csv**.

```
# Read csv file into a pandas dataframe
df = pd.read_csv("E:/property.csv")
```

```
df
```

	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	104.0	PUTNAM	Y	3	1	1000
1	197.0	LEXINGTON	N	3	1.5	--
2	NaN	LEXINGTON	N	NaN	1	850
3	201.0	BERKELEY	12	1	NaN	700
4	203.0	BERKELEY	Y	3	2	1600
5	207.0	BERKELEY	Y	NaN	1	800
6	NaN	WASHINGTON	NaN	2	--	950
7	213.0	TREMONT	Y	--	1	NaN
8	215.0	TREMONT	Y	na	2	1800

Check for Missing Values

Going back to our original dataset, let's take a look at the "ST_NUM" column.

In the third row there is an empty cell and seventh row there's an "NA" value.

Clearly these are both missing values. Let's see how Pandas deals with these.

```
df['ST_NUM']
```

```
0    104.0
1    197.0
2      NaN
3    201.0
4    203.0
5    207.0
6      NaN
7    213.0
8    215.0
Name: ST_NUM, dtype: float64
```

```
# Looking at the ST_NUM column for null values
df['ST_NUM'].isnull()
```

```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7    False
8    False
Name: ST_NUM, dtype: bool
```

Taking a look at the column, we can see that Pandas filled in the blank space with “NA”. Using the isnull() method, we can confirm that both the missing value and “NA” were recognized as missing values. Both boolean responses are True. This is a simple example, but highlights an important point. Pandas will recognize both empty cells and “NA” types as missing values.

Let’s take a look at the “NUM_BEDROOMS” column.

Sometimes it might be the case where there’s missing values that have different formats.
In this column, there’s four missing values.

- n/a
- NA
- —
- na

Just like before, Pandas recognized the “NA” as a missing value.

```
# Looking at the NUM_BEDROOMS column
df['NUM_BEDROOMS']
```

```
0      3
1      3
2    NaN
3      1
4      3
5    NaN
6      2
7     --
8     na
Name: NUM_BEDROOMS, dtype: object
```

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

```
0    False
1    False
2     True
3    False
4    False
5     True
6    False
7    False
8    False
Name: NUM_BEDROOMS, dtype: bool
```

If there's multiple users manually entering data, then this is a common problem. Maybe i like to use "n/a" but you like to use "na".

An easy way to detect these various formats is to put them in a list. Then when we import the data, Pandas will recognize them right away.

```
# Making a list of missing value types
missing_values = ["n/a", "na", "--"]
df = pd.read_csv("E:/PES/IDSDATASET/property.csv", na_values = missing_values)
df
```

	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	104.0	PUTNAM	Y	3.0	1.0	1000.0
1	197.0	LEXINGTON	N	3.0	1.5	NaN
2	NaN	LEXINGTON	N	NaN	1.0	850.0
3	201.0	BERKELEY	12	1.0	NaN	700.0
4	203.0	BERKELEY	Y	3.0	2.0	1600.0
5	207.0	BERKELEY	Y	NaN	1.0	800.0
6	NaN	WASHINGTON	NaN	2.0	NaN	950.0
7	213.0	TREMONT	Y	NaN	1.0	NaN
8	215.0	TREMONT	Y	NaN	2.0	1800.0

:
```
# Looking at the NUM_BEDROOMS column
df['NUM_BEDROOMS']
```

:
```
0    3.0
1    3.0
2    NaN
3    1.0
4    3.0
5    NaN
6    2.0
7    NaN
8    NaN
Name: NUM_BEDROOMS, dtype: float64
```

:
```
df['NUM_BEDROOMS'].isnull()
```

:
```
0    False
1    False
2     True
3    False
4    False
5     True
6    False
7     True
8     True
Name: NUM_BEDROOMS, dtype: bool
```

This time, all of the different formats were recognized as missing values.

Let's take a look at the "Owner Occupied" column.

```
# Looking at the OWN_OCCUPIED column
df['OWN_OCCUPIED']
```

```
0      Y
1      N
2      N
3     12
4      Y
5      Y
6    NaN
7      Y
8      Y
Name: OWN_OCCUPIED, dtype: object
```

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

```
0    False
1    False
2    False
3    False
4    False
5    False
6     True
7    False
8    False
Name: OWN_OCCUPIED, dtype: bool
```

In the fourth row, there's the number 12. The response for Owner Occupied should clearly be a string (Y or N), so this numeric type should be a missing value.

1. Loop through the OWN_OCCUPIED column
2. Try and turn the entry into an integer
3. If the entry can be changed into an integer, enter a missing value
4. If the number can't be an integer, we know it's a string, so keep going

```
# Detecting numbers
cnt=0
for row in df['OWN_OCCUPIED']:
    try:
        int(row)
        df.loc[cnt, 'OWN_OCCUPIED']=np.nan
    except ValueError:
        pass
    cnt+=1
```

Summarizing Missing Values

After we've cleaned the missing values, we will probably want to summarize them. For instance, we might want to look at the total number of missing values for each feature.

```
# Total missing values for each feature
df.isnull().sum()
```

```
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    2
NUM_BEDROOMS    4
NUM_BATH        2
SQ_FT           2
dtype: int64
```

```
# Any missing values?
df.isnull().values.any()
```

```
True
```

```
# Total number of missing values
df.isnull().sum().sum()
```

```
12
```

Replacing

Fill in missing values with a single value.

```
# Replace missing values with a number
df['ST_NUM'].fillna(125, inplace=True)
```

```
df
```

	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	104.0	PUTNAM	Y	3.0	1.0	1000.0
1	197.0	LEXINGTON	N	3.0	1.5	NaN
2	125.0	LEXINGTON	N	NaN	1.0	850.0
3	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	203.0	BERKELEY	Y	3.0	2.0	1600.0
5	207.0	BERKELEY	Y	NaN	1.0	800.0
6	125.0	WASHINGTON	NaN	2.0	NaN	950.0
7	213.0	TREMONT	Y	NaN	1.0	NaN
8	215.0	TREMONT	Y	NaN	2.0	1800.0

```
# Location based replacement
df.loc[2,'ST_NUM'] = 125
```

```
df
```

	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	104.0	PUTNAM	Y	3.0	1.0	1000.0
1	197.0	LEXINGTON	N	3.0	1.5	NaN
2	125.0	LEXINGTON	N	NaN	1.0	850.0
3	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	203.0	BERKELEY	Y	3.0	2.0	1600.0
5	207.0	BERKELEY	Y	NaN	1.0	800.0
6	125.0	WASHINGTON	NaN	2.0	NaN	950.0
7	213.0	TREMONT	Y	NaN	1.0	NaN
8	215.0	TREMONT	Y	NaN	2.0	1800.0

```
# Replace using median - Bedrooms
median = df['NUM_BEDROOMS'].median()
df['NUM_BEDROOMS'].fillna(median, inplace=True)
```

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1.0	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	125.0	LEXINGTON	N	3.0	1.0	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2.0	1600.0
5	100006000.0	207.0	BERKELEY	Y	3.0	1.0	800.0
6	100007000.0	125.0	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	3.0	1.0	NaN
8	100009000.0	215.0	TREMONT	Y	3.0	2.0	1800.0

Source: https://towardsdatascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b