**Aim**: Write a code to read data from the different file formats like JSON, HTML, XML, and CSV files and check for missing data and outlier values and handle them.

**Theory:**

Data is an important factor in all domains, industries, education, research, etc. And the data is not present in the same format. Even in a domain, we deal with different forms of information. In some cases, it may even be unstructured. Python being versatile, does have its hand in accessing various file formats. In this article, we will be discussing how to handle the most common file formats, namely, Text, CSV, XLSX, JSON, etc. using Python.

**Code:**

# read csv file

import pandas as pd

df = pd.read\_csv(r'property.csv')

df

#read xml file

df = pd.read\_xml('sample.xml')

df.head()

# read Excel file into a DataFrame

df = pd.read\_excel(r'file\_example\_XLSX\_50.xlsx')

# print values

df

# read Json file

import json

with open('comments.json') as data:

     JSONdta = json.load(data)

print(JSONdta)

# Importing libraries

import pandas as pd

import numpy as np

# Read csv file into a pandas dataframe

df = pd.read\_csv("property.csv")

# Read csv file into a pandas dataframe

df = pd.read\_csv("property.csv")

# Take a look at the first few rows

df.head()

# Standard Missing Values

# Looking at the ST\_NUM column

df['ST\_NUM']

df['ST\_NUM'].isnull()

# Non-Standard Missing Values

# Looking at the NUM\_BEDROOMS column

df['NUM\_BEDROOMS']

df['NUM\_BEDROOMS'].isnull()

# Replace missing values with a number

df['ST\_NUM'].fillna(125, inplace=True)

# Location based replacement

df.loc[2,'ST\_NUM'] = 125

df

# Import required libraries

# Importing

import sklearn

from sklearn.datasets import load\_boston

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the dataset

bos\_hou = load\_boston()

# Create the dataframe

column\_name = bos\_hou.feature\_names

df\_boston = pd.DataFrame(bos\_hou.data)

df\_boston.columns = column\_name

df\_boston.head()

df\_boston.describe()

# Detecting the outliers using IQR and removing them.

# Importing

import sklearn

from sklearn.datasets import load\_boston

import pandas as pd

# Load the dataset

bos\_hou = load\_boston()

# Create the dataframe

column\_name = bos\_hou.feature\_names

df\_boston = pd.DataFrame(bos\_hou.data)

df\_boston.columns = column\_name

df\_boston.head()

''' Detection '''

# IQR

Q1 = np.percentile(df\_boston['DIS'], 25,

                   interpolation = 'midpoint')

Q3 = np.percentile(df\_boston['DIS'], 75,

                   interpolation = 'midpoint')

IQR = Q3 - Q1

print("Old Shape: ", df\_boston.shape)

# Upper bound

upper = np.where(df\_boston['DIS'] >= (Q3+1.5\*IQR))

# Lower bound

lower = np.where(df\_boston['DIS'] <= (Q1-1.5\*IQR))

''' Removing the Outliers '''

df\_boston.drop(upper[0], inplace = True)

df\_boston.drop(lower[0], inplace = True)

print("New Shape: ", df\_boston.shape)

# Visualization

# 1. Box Plot

# Box Plot

import seaborn as sns

sns.boxplot(df\_boston['DIS'])

# Position of the Outlier

print(np.where(df\_boston['DIS']>10))

(array([351, 352, 353, 354, 355]),)

# 2. Histogram

df\_boston.DIS.hist()

# 3. Scatterplot

# Scatter plot

fig, ax = plt.subplots(figsize = (18,10))

ax.scatter(df\_boston['INDUS'], df\_boston['TAX'])

# x-axis label

ax.set\_xlabel('(Proportion non-retail business acres)/(town)')

# y-axis label

ax.set\_ylabel('(Full-value property-tax rate)/( $10,000)')

plt.show()

# Position of the Outlier

print(np.where((df\_boston['INDUS']>20) & (df\_boston['TAX']>600)))

**Output :**

# read csv file

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PID** | **ST\_NUM** | **ST\_NAME** | **OWN\_OCCUPIED** | **NUM\_BEDROOMS** | **NUM\_BATH** | **SQ\_FT** |
| **0** | 100001000.0 | 104.0 | PUTNAM | Y | 3 | 1 | 1000 |
| **1** | 100002000.0 | 197.0 | LEXINGTON | N | 3 | 1.5 | -- |
| **2** | 100003000.0 | NaN | LEXINGTON | N | NaN | 1 | 850 |
| **3** | 100004000.0 | 201.0 | BERKELEY | 12 | 1 | NaN | 700 |
| **4** | NaN | 203.0 | BERKELEY | Y | 3 | 2 | 1600 |
| **5** | 100006000.0 | 207.0 | BERKELEY | Y | NaN | 1 | 800 |
| **6** | 100007000.0 | NaN | WASHINGTON | NaN | 2 | HURLEY | 950 |
| **7** | 100008000.0 | 213.0 | TREMONT | Y | 1 | 1 | NaN |
| **8** | 100009000.0 | 215.0 | TREMONT | Y | na | 2 | 1800 |

#read xml file

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **id** | **author** | **title** | **genre** | **price** | **publish\_date** | **description** |
| **0** | bk101 | Gambardella, Matthew | XML Developer's Guide | Computer | 44.95 | 2000-10-01 | An in-depth look at creating applications \n ... |
| **1** | bk102 | Ralls, Kim | Midnight Rain | Fantasy | 5.95 | 2000-12-16 | A former architect battles corporate zombies, ... |
| **2** | bk103 | Corets, Eva | Maeve Ascendant | Fantasy | 5.95 | 2000-11-17 | After the collapse of a nanotechnology \n ... |
| **3** | bk104 | Corets, Eva | Oberon's Legacy | Fantasy | 5.95 | 2001-03-10 | In post-apocalypse England, the mysterious \n ... |
| **4** | bk105 | Corets, Eva | The Sundered Grail | Fantasy | 5.95 | 2001-09-10 | The two daughters of Maeve, half-sisters, \n ... |

# read Excel file into a DataFrame

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **First Name** | **Last Name** | **Gender** | **Country** | **Age** | **Date** | **Id** |
| **0** | 1 | Dulce | Abril | Female | United States | 32 | 15/10/2017 | 1562 |
| **1** | 2 | Mara | Hashimoto | Female | Great Britain | 25 | 16/08/2016 | 1582 |
| **2** | 3 | Philip | Gent | Male | France | 36 | 21/05/2015 | 2587 |
| **3** | 4 | Kathleen | Hanner | Female | United States | 25 | 15/10/2017 | 3549 |
| **4** | 5 | Nereida | Magwood | Female | United States | 58 | 16/08/2016 | 2468 |
| **5** | 6 | Gaston | Brumm | Male | United States | 24 | 21/05/2015 | 2554 |
| **6** | 7 | Etta | Hurn | Female | Great Britain | 56 | 15/10/2017 | 3598 |

# read Json file

{'note': 'This file contains the actual data for your assignment', 'comments': [{'name': 'Abaan', 'count': 98}, {'name': 'Ashna', 'count': 95}, {'name': 'Dante', 'count': 94}, {'name': 'Isabel', 'count': 93}, {'name': 'Fearne', 'count': 92}, {'name': 'Kriss', 'count': 91}, {'name': 'Janani', 'count': 87}, {'name': 'Karhys', 'count': 85}, {'name': 'Megg', 'count': 84}, {'name': 'Luisa', 'count': 83}, {'name': 'Thorben', 'count': 79}, {'name': 'Kaelan', 'count': 77}, {'name': 'Ceirin', 'count': 75}, {'name': 'Lileidh', 'count': 70}, {'name': 'Angelika', 'count': 70}, {'name': 'Amelka', 'count': 69}, {'name': 'Justin', 'count': 69}, {'name': 'Muneeb', 'count': 68}, {'name': 'Antoine', 'count': 64}, {'name': 'Ivar', 'count': 61}, {'name': 'Kaid', 'count': 60}, {'name': 'Dakotah', 'count': 58}, {'name': 'Nadeem', 'count': 58}, {'name': 'Marybeth', 'count': 55}, {'name': 'Ashlyn', 'count': 55}, {'name': 'Kaydin', 'count': 50}, {'name': 'Obieluem', 'count': 48}, {'name': 'Cairn', 'count': 46}, {'name': 'Ala', 'count': 45}, {'name': 'Vithujan', 'count': 38}, {'name': 'Ivory', 'count': 34}, {'name': 'Rosalyn', 'count': 33}, {'name': 'Kaywan', 'count': 32}, {'name': 'Pedro', 'count': 31}, {'name': 'Bharath', 'count': 30}, {'name': 'Eshaal', 'count': 29}, {'name': 'Aliya', 'count': 28}, {'name': 'Sephiroth', 'count': 27}, {'name': 'Minah', 'count': 25}, {'name': 'Murdo', 'count': 22}, {'name': 'Ata', 'count': 21}, {'name': 'Remonae', 'count': 17}, {'name': 'Muskaan', 'count': 17}, {'name': 'Lottie', 'count': 17}, {'name': 'Giane', 'count': 9}, {'name': 'Dineo', 'count': 6}, {'name': 'Zoe', 'count': 5}, {'name': 'Raul', 'count': 4}, {'name': 'Tammylee', 'count': 2}, {'name': 'Morna', 'count': 1}]}

# Take a look at the first few rows

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PID** | **ST\_NUM** | **ST\_NAME** | **OWN\_OCCUPIED** | **NUM\_BEDROOMS** | **NUM\_BATH** | **SQ\_FT** |
| **0** | 100001000.0 | 104.0 | PUTNAM | Y | 3 | 1 | 1000 |
| **1** | 100002000.0 | 197.0 | LEXINGTON | N | 3 | 1.5 | -- |
| **2** | 100003000.0 | NaN | LEXINGTON | N | NaN | 1 | 850 |
| **3** | 100004000.0 | 201.0 | BERKELEY | 12 | 1 | NaN | 700 |
| **4** | NaN | 203.0 | BERKELEY | Y | 3 | 2 | 1600 |

# Standard Missing Values

# Looking at the ST\_NUM column

0    False

1    False

2     True

3    False

4    False

5    False

6     True

7    False

8    False

Name: ST\_NUM, dtype: bool

# Non-Standard Missing Values

# Looking at the NUM\_BEDROOMS column

0    False

1    False

2     True

3    False

4    False

5     True

6    False

7    False

8    False

Name: NUM\_BEDROOMS, dtype: bool

# Replace missing values with a number

# Location based replacement

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PID** | **ST\_NUM** | **ST\_NAME** | **OWN\_OCCUPIED** | **NUM\_BEDROOMS** | **NUM\_BATH** | **SQ\_FT** |
| **0** | 100001000.0 | 104.0 | PUTNAM | Y | 3 | 1 | 1000 |
| **1** | 100002000.0 | 197.0 | LEXINGTON | N | 3 | 1.5 | -- |
| **2** | 100003000.0 | 125.0 | LEXINGTON | N | NaN | 1 | 850 |
| **3** | 100004000.0 | 201.0 | BERKELEY | 12 | 1 | NaN | 700 |
| **4** | NaN | 203.0 | BERKELEY | Y | 3 | 2 | 1600 |
| **5** | 100006000.0 | 207.0 | BERKELEY | Y | NaN | 1 | 800 |
| **6** | 100007000.0 | 125.0 | WASHINGTON | NaN | 2 | HURLEY | 950 |
| **7** | 100008000.0 | 213.0 | TREMONT | Y | 1 | 1 | NaN |
| **8** | 100009000.0 | 215.0 | TREMONT | Y | na | 2 | 1800 |

handle Outlier

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CRIM** | **ZN** | **INDUS** | **CHAS** | **NOX** | **RM** | **AGE** | **DIS** | **RAD** | **TAX** | **PTRATIO** | **B** | **LSTAT** |
| **0** | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396.90 | 4.98 |
| **1** | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 |
| **2** | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 |
| **3** | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CRIM** | **ZN** | **INDUS** | **CHAS** | **NOX** | **RM** | **AGE** | **DIS** | **RAD** | **TAX** | **PTRATIO** | **B** | **LSTAT** |
| **count** | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 |
| **mean** | 3.613524 | 11.363636 | 11.136779 | 0.069170 | 0.554695 | 6.284634 | 68.574901 | 3.795043 | 9.549407 | 408.237154 | 18.455534 | 356.674032 | 12.653063 |
| **std** | 8.601545 | 23.322453 | 6.860353 | 0.253994 | 0.115878 | 0.702617 | 28.148861 | 2.105710 | 8.707259 | 168.537116 | 2.164946 | 91.294864 | 7.141062 |
| **min** | 0.006320 | 0.000000 | 0.460000 | 0.000000 | 0.385000 | 3.561000 | 2.900000 | 1.129600 | 1.000000 | 187.000000 | 12.600000 | 0.320000 | 1.730000 |
| **25%** | 0.082045 | 0.000000 | 5.190000 | 0.000000 | 0.449000 | 5.885500 | 45.025000 | 2.100175 | 4.000000 | 279.000000 | 17.400000 | 375.377500 | 6.950000 |
| **50%** | 0.256510 | 0.000000 | 9.690000 | 0.000000 | 0.538000 | 6.208500 | 77.500000 | 3.207450 | 5.000000 | 330.000000 | 19.050000 | 391.440000 | 11.360000 |
| **75%** | 3.677083 | 12.500000 | 18.100000 | 0.000000 | 0.624000 | 6.623500 | 94.075000 | 5.188425 | 24.000000 | 666.000000 | 20.200000 | 396.225000 | 16.955000 |
| **max** | 88.976200 | 100.000000 | 27.740000 | 1.000000 | 0.871000 | 8.780000 | 100.000000 | 12.126500 | 24.000000 | 711.000000 | 22.000000 | 396.900000 | 37.970000 |

Old Shape:  (506, 13)

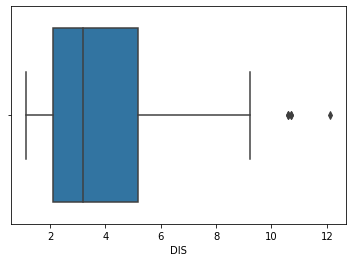
New Shape:  (501, 13)

# Visualization

# 1. Box Plot

# Box Plot

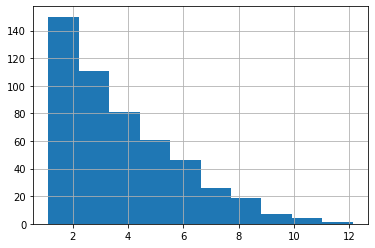
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f5803a19e50>



# Position of the Outlier

# 2. Histogram

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f58036c1550>



**Result:**

Here in this practical we have use various like excel, csv, json file and also handled the outliers in the data.