Outlier Treatment

Outlier analysis is the process of identifying outliers, or abnormal observations, in a dataset. Also known as outlier detection, it's an important step in data analysis, as it removes erroneous or inaccurate observations which might otherwise skew conclusions.

In [1]:
```
# Importing the packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Loading the data
df = pd.read_csv(r"F:\022 Study Material\Assignments & Keys\Assignments - Keys\Data
df
```

Out[2]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	
0	0.15876	0.0	10.81	0.0	0.413	5.961	17.5	5.2873	4.0	305.0	19.2	376.94	9.88	
1	0.10328	25.0	5.13	0.0	0.453	5.927	47.2	6.9320	8.0	284.0	19.7	396.90	9.22	
2	0.34940	0.0	9.90	0.0	0.544	5.972	76.7	3.1025	4.0	304.0	18.4	396.24	9.97	
3	2.73397	0.0	19.58	0.0	0.871	5.597	94.9	1.5257	5.0	403.0	14.7	351.85	21.45	
4	0.04337	21.0	5.64	0.0	0.439	6.115	63.0	6.8147	4.0	243.0	16.8	393.97	9.43	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
399	9.32909	0.0	18.10	0.0	0.713	6.185	98.7	2.2616	24.0	666.0	20.2	396.90	18.13	
400	51.13580	0.0	18.10	0.0	0.597	5.757	100.0	1.4130	24.0	666.0	20.2	2.60	10.11	
401	0.01501	90.0	1.21	1.0	0.401	7.923	24.8	5.8850	1.0	198.0	13.6	395.52	3.16	
402	0.02055	85.0	0.74	0.0	0.410	6.383	35.7	9.1876	2.0	313.0	17.3	396.90	5.77	
403	0.08244	30.0	4.93	0.0	0.428	6.481	18.5	6.1899	6.0	300.0	16.6	379.41	6.36	

404 rows × 14 columns

In [3]:
```
# Checking the list of columns in data
df.columns
```

Out[3]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

In [4]:
```
# Finding data types
df.dtypes
```

Out[4]:
```
crim       float64
zn         float64
indus      float64
chas       float64
nox        float64
rm         float64
age        float64
dis        float64
rad        float64
tax        float64
ptratio    float64
black      float64
lstat      float64
medv       float64
dtype: object
```

In [5]:
```
# Checking if there are any Null values
df.isna().sum()
```

Out[5]:
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

In [6]:
```
# Checking the descriptive statistics
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
crim	404.0	3.730912	8.943922	0.00632	0.082382	0.253715	4.053158	88.9762
zn	404.0	10.509901	22.053733	0.00000	0.000000	0.000000	12.500000	95.0000
indus	404.0	11.189901	6.814909	0.46000	5.190000	9.795000	18.100000	27.7400
chas	404.0	0.069307	0.254290	0.00000	0.000000	0.000000	0.000000	1.0000
nox	404.0	0.556710	0.117321	0.39200	0.453000	0.538000	0.631000	0.8710
rm	404.0	6.301450	0.675830	3.56100	5.902750	6.230500	6.629250	8.7800
age	404.0	68.601733	28.066143	2.90000	45.800000	76.600000	94.150000	100.0000
dis	404.0	3.799666	2.109916	1.16910	2.087875	3.207450	5.222125	12.1265
rad	404.0	9.836634	8.834741	1.00000	4.000000	5.000000	24.000000	24.0000
tax	404.0	411.688119	171.073553	187.00000	281.000000	330.000000	666.000000	711.0000
ptratio	404.0	18.444554	2.150295	12.60000	17.375000	19.000000	20.200000	22.0000
black	404.0	355.068243	94.489572	0.32000	374.710000	391.065000	396.007500	396.9000
lstat	404.0	12.598936	6.925173	1.73000	7.135000	11.265000	16.910000	34.3700
medv	404.0	22.312376	8.837019	5.00000	17.100000	21.400000	25.000000	50.0000

In [7]:
```
# define the columns you want to check for outliers
cols = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'p
```

In [8]:
```
# create an empty dictionary to store the results
outliers = {}
```

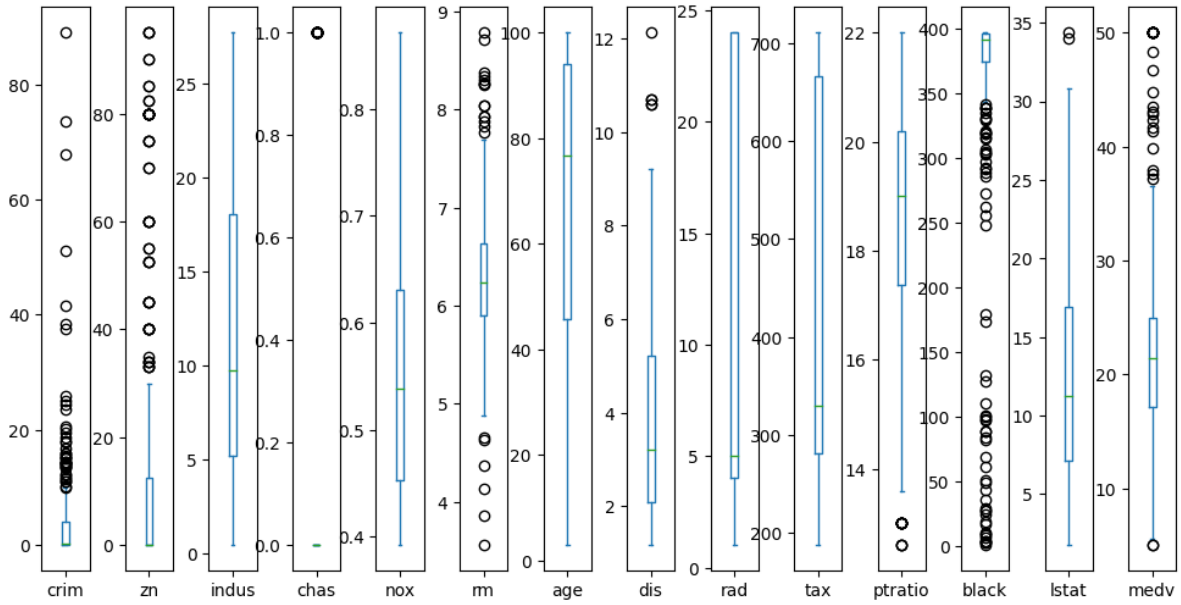
In [9]:
```
# loop through each column and calculate the number of outliers
for col in cols:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5*iqr
    upper_bound = q3 + 1.5*iqr
    outliers_count = len(df[(df[col] < lower_bound) | (df[col] > upper_bound)])
    if outliers_count > 0:
        outliers[col] = outliers_count
```

In [10]:
```
# print the results
if len(outliers) == 0:
    print("No outliers found")
else:
    for col, count in outliers.items():
        print(f"Column '{col}' has {count} outliers")
```

```
Column 'crim' has 42 outliers
Column 'zn' has 49 outliers
Column 'chas' has 28 outliers
Column 'rm' has 21 outliers
Column 'dis' has 5 outliers
Column 'ptratio' has 12 outliers
Column 'black' has 61 outliers
Column 'lstat' has 2 outliers
Column 'medv' has 27 outliers
```

In [11]:
```
# Multiple boxplots in a single visualization.
df.plot(kind = 'box', subplots = True, sharey = False, figsize = (12, 6))

# increase spacing between subplots
plt.subplots_adjust(wspace = 0.75)
plt.show()
```



In [12]:
```
from feature_engine.outliers import Winsorizer
from sklearn.compose import ColumnTransformer
```

In [13]:
```
# creating the Winsorizer with the IQR method
winsor_IQR = Winsorizer(capping_method = 'iqr',
                        tail = 'both',
                        fold = 1.5,
                        variables = ['crim', 'zn', 'indus', 'nox',
                                     'rm', 'age', 'dis', 'rad', 'tax',
                                     'ptratio', 'black', 'lstat', 'medv'])
```

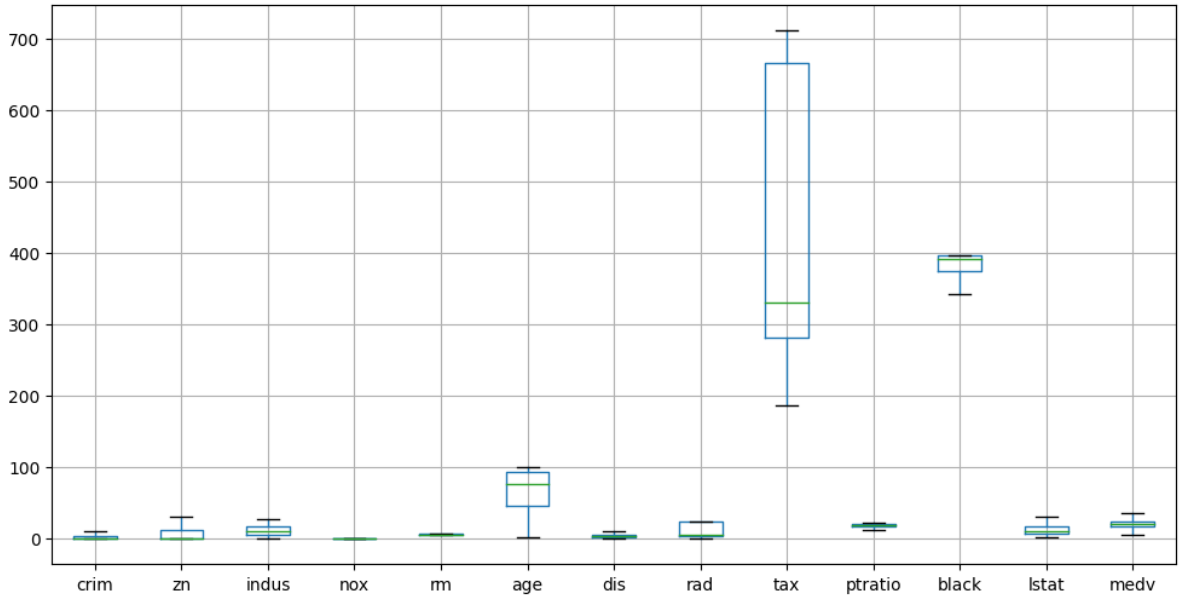
In [14]:
```
winsor_IQR
```

Out[14]:
```
Winsorizer(capping_method='iqr', fold=1.5, tail='both',
           variables=['crim', 'zn', 'indus', 'nox', 'rm', 'age', 'dis',
'rad',
                      'tax', 'ptratio', 'black', 'lstat', 'medv'])
```

In [15]:
```
# fitting the Winsorizer to the dataframe
clean_df = winsor_IQR.fit_transform(df)
```

In [16]:
```
# checking for outliers after applying the Winsorizer
clean_df.boxplot(column = ['crim', 'zn', 'indus', 'nox',
                           'rm', 'age', 'dis', 'rad', 'tax',
                           'ptratio', 'black', 'lstat', 'medv'],
                 figsize = (12, 6))
```

Out[16]:
```
<AxesSubplot: >
```


700
600
500
400
300
200
100
0
crim
zn
indus
nox
rm
age
dis
rad
tax
ptratio
black
lstat
medv

In [ ]: