Remove_Outliers

March 29, 2023

[611]:
```
1#Import pandas and matplotlib
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
from scipy import stats
import seaborn as sns
```

[612]:
```
2#test csv
df=pd.read_csv ('artificial_housing_data.csv')
df.head()
```

[612]:

	timestamp	house_id	house_size	temperature	humidity	precipitation	wind_speed	solar_irradiance	week_of_the_year	hour_of_the_day	day_of_the_week	wday__0	wday__1	wday__2	wday__3	wday__4	wday__5	wday__6	consumption
0	2014-12-01 00:00:00+00:00	house_0	38.4	12.856301	0.227995	1.272970	4.708708	195.588130	49	0	0	1	0	0	0	0	0	0	77.184583
1	2014-12-01 01:00:00+00:00	house_0	38.4	12.126253	0.220464	2.814100	3.156671	203.112272	49	1	0	1	0	0	0	0	0	0	72.301535
2	2014-12-01 02:00:00+00:00	house_0	38.4	15.277380	0.361294	0.515568	3.490243	204.453061	49	2	0	1	0	0	0	0	0	0	68.666249
3	2014-12-01 03:00:00+00:00	house_0	38.4	12.817692	0.236057	1.920968	3.191047	192.645952	49	3	0	1	0	0	0	0	0	0	
4	2014-12-01 04:00:00+00:00	house_0	38.4	14.343213	0.051376	2.292260	3.142263	196.727594	49	4	0	1	0	0	0	0	0	0	

```
3        0        0        0    58.396602
4        0        0        0    76.230163
```

[613]:
```
3#check the shape of dataFrame
df.shape
```

[613]:
```
(218925, 19)
```

[614]:
```
4#descriptive_state
df.describe
```

[614]:
```
<bound method NDFrame.describe of                          timestamp  house_id
house_size  temperature  \
0       2014-12-01 00:00:00+00:00   house_0      38.400    12.856301
1       2014-12-01 01:00:00+00:00   house_0      38.400    12.126253
2       2014-12-01 02:00:00+00:00   house_0      38.400    15.277380
3       2014-12-01 03:00:00+00:00   house_0      38.400    12.817692
4       2014-12-01 04:00:00+00:00   house_0      38.400    14.343213
…                             …       …          …           …
218920  2015-11-30 19:00:00+00:00  house_24      50.025     8.784783
218921  2015-11-30 20:00:00+00:00  house_24      50.025     7.578867
218922  2015-11-30 21:00:00+00:00  house_24      50.025     6.030384
218923  2015-11-30 22:00:00+00:00  house_24      50.025     8.921952
218924  2015-11-30 23:00:00+00:00  house_24      50.025     6.267010

        humidity  precipitation  wind_speed  solar_irradiance  \
0       0.227995       1.272970    4.708708        195.588130
1       0.220464       2.814100    3.156671        203.112272
2       0.361294       0.515568    3.490243        204.453061
3       0.236057       1.920968    3.191047        192.645952
4       0.051376       2.292260    3.142263        196.727594
…            …              …           …                 …
218920  0.187472       1.499202    3.123499        197.511497
218921  0.120572       0.003390    2.539654        194.567294
218922  0.290202       2.695812    3.646082        193.759270
218923  0.241852       0.987437    3.273055        193.294319
218924  0.007762       2.824886    3.344415        197.644169

        week_of_the_year  hour_of_the_day  day_of_the_week  wday__0  wday__1  \
0                     49                0                0        1        0
1                     49                1                0        1        0
2                     49                2                0        1        0
3                     49                3                0        1        0
4                     49                4                0        1        0
…                    …                 …                …        …        …
218920                49               19                0        1        0
218921                49               20                0        1        0
```

```
218922              49               21               0        1        0
218923              49               22               0        1        0
218924              49               23               0        1        0

        wday__2  wday__3  wday__4  wday__5  wday__6  consumption
0             0        0        0        0        0    77.184583
1             0        0        0        0        0    72.301535
2             0        0        0        0        0    68.666249
3             0        0        0        0        0    58.396602
4             0        0        0        0        0    76.230163
…           …        …        …        …        …           …
218920        0        0        0        0        0   101.813038
218921        0        0        0        0        0    99.244315
218922        0        0        0        0        0    83.341239
218923        0        0        0        0        0    91.493841
218924        0        0        0        0        0    74.526816

[218925 rows x 19 columns]>
```

[615]:
```
5#delete unsed data from test
df.drop('timestamp', inplace=True, axis=1)
df.drop('hour_of_the_day', inplace=True, axis=1)
df.drop('day_of_the_week', inplace=True, axis=1)
df.drop('wday__0', inplace=True, axis=1)
df.drop('wday__1', inplace=True, axis=1)
df.drop('wday__2', inplace=True, axis=1)
df.drop('wday__3', inplace=True, axis=1)
df.drop('wday__4', inplace=True, axis=1)
df.drop('wday__5', inplace=True, axis=1)
df.drop('wday__6', inplace=True, axis=1)
df.drop('house_id', inplace=True, axis=1)
df.drop('house_size', inplace=True, axis=1)
df.drop('humidity', inplace=True, axis=1)
df.drop('week_of_the_year', inplace=True, axis=1)
```

[616]:
```
df.describe
```

[616]:
```
<bound method NDFrame.describe of         temperature  precipitation  wind_speed
solar_irradiance  consumption
0         12.856301       1.272970    4.708708        195.588130    77.184583
1         12.126253       2.814100    3.156671        203.112272    72.301535
2         15.277380       0.515568    3.490243        204.453061    68.666249
3         12.817692       1.920968    3.191047        192.645952    58.396602
4         14.343213       2.292260    3.142263        196.727594    76.230163
…               …              …           …                 …           …
218920     8.784783       1.499202    3.123499        197.511497   101.813038
218921     7.578867       0.003390    2.539654        194.567294    99.244315
```

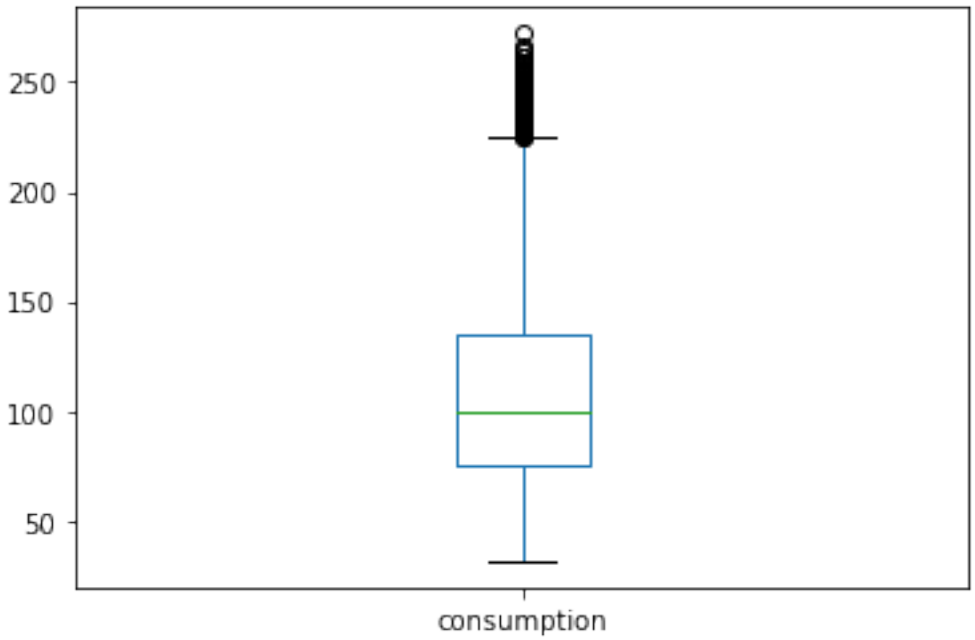
```
218922     6.030384       2.695812    3.646082       193.759270    83.341239
218923     8.921952       0.987437    3.273055       193.294319    91.493841
218924     6.267010       2.824886    3.344415       197.644169    74.526816

[218925 rows x 5 columns]>
```

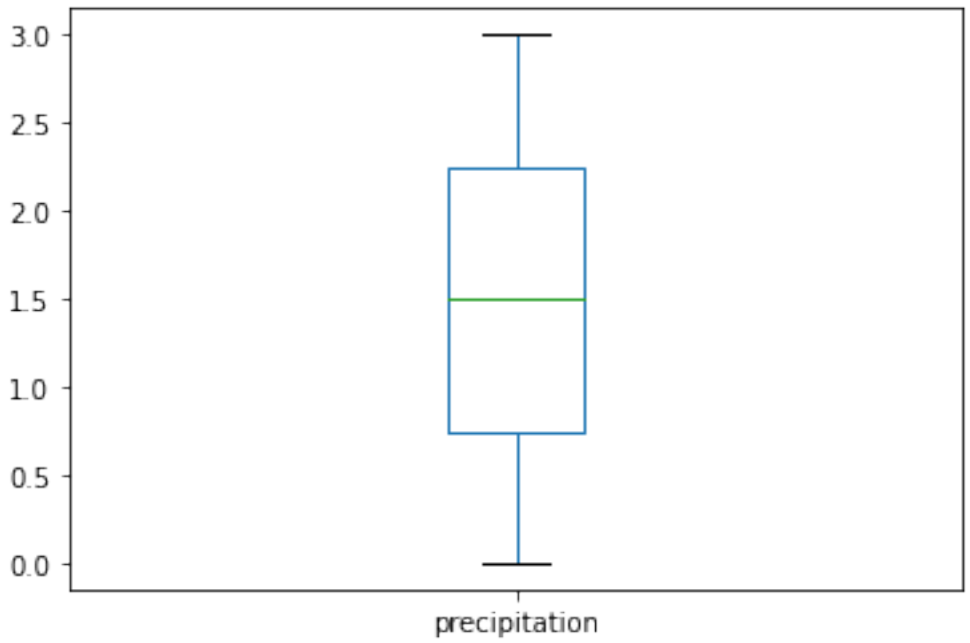
[617]:
```
6#define function called "plot_boxplot"

def plot_boxplot(df,ft):
    df.boxplot(column=[ft])
    plt.grid(False)
    plt.show()
```

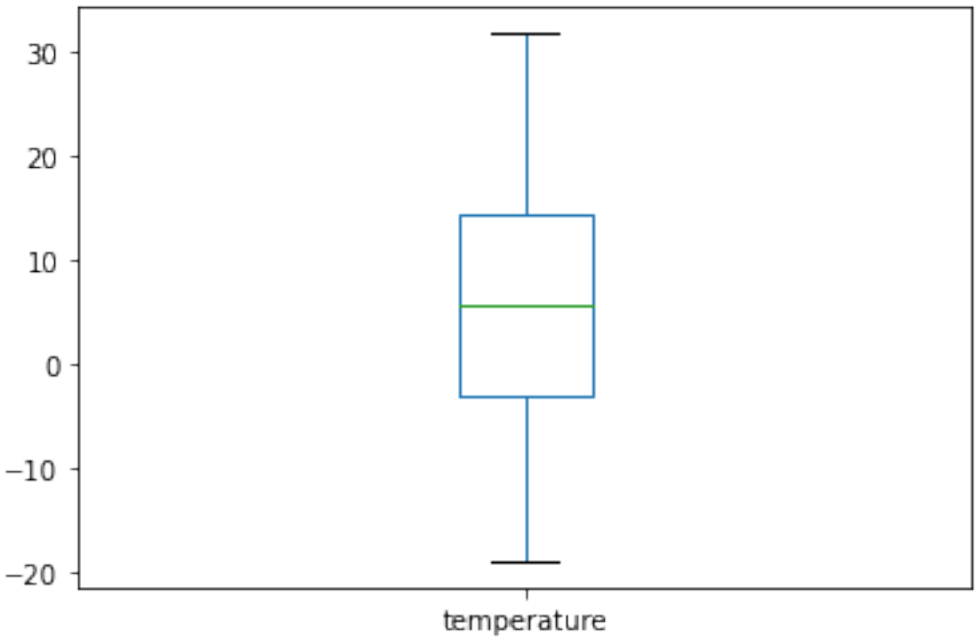
[618]:
```
7#first Feature consumption
plot_boxplot(df,"consumption")
```



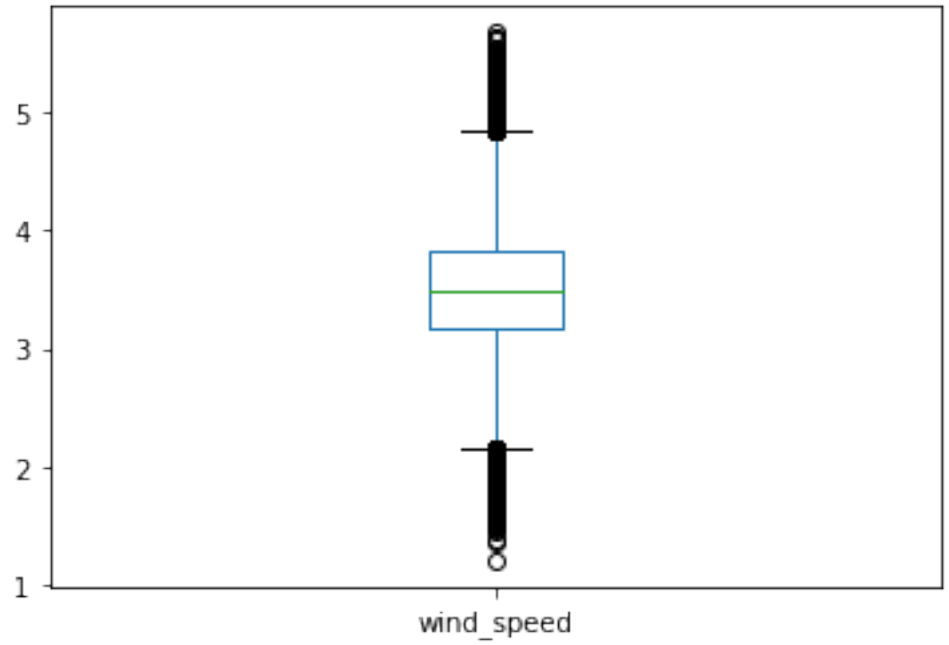
[619]:
```
#second Feature precipitation
plot_boxplot(df,"precipitation")
```



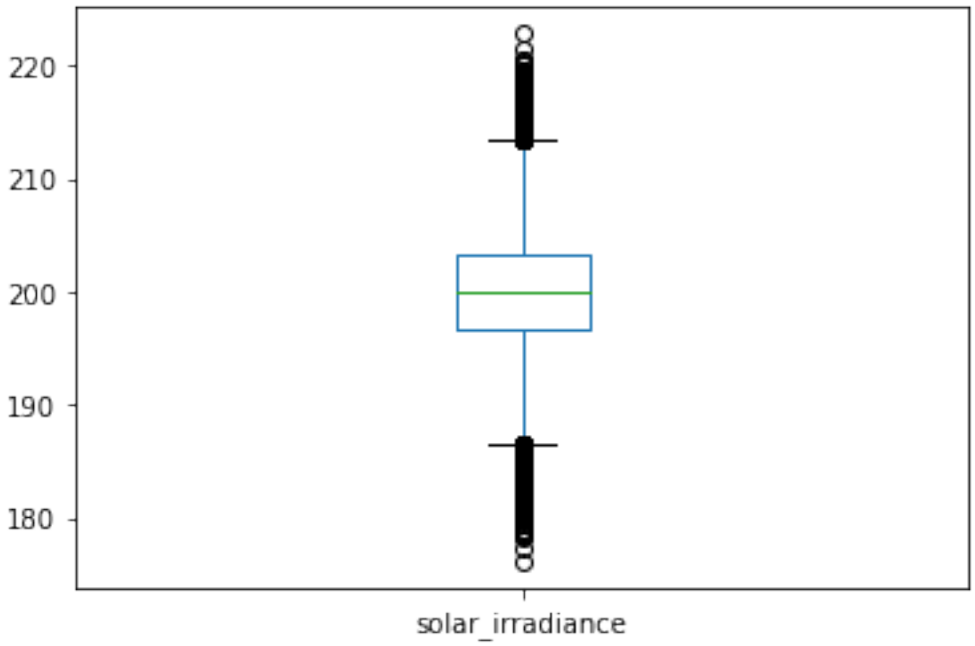
[620]:
```
#Third Feature Tempreture
plot_boxplot(df,"temperature")
```



[621]:
```
#Fourth Feature wind_speed
plot_boxplot(df,"wind_speed")
```



[622]:
```
#Fifth Feature solar_irradiance
plot_boxplot(df,"solar_irradiance")
```



[623]:
```
8#define a function called "outliers" returns a list of index of outliers
#IQR =Q3-Q1
# +/- 1.5* IQR

def outliers (df,ft):
    Q1=df[ft].quantile(0.25)
    Q3=df[ft].quantile(0.75)
    IQR=Q3-Q1

    Lower_bound=Q1-1.5*IQR
    Upper_bound=Q3+1.5*IQR

    ls=df.index[ (df[ft]< Lower_bound) | (df[ft] > Upper_bound)]

    return ls
```

[624]:
```
9# create an empty List to store the outliers from multiple columns

index_list=[]
for feature in ["temperature" ,"wind_speed",␣
 ↪"solar_irradiance","consumption","precipitation"]:

    index_list.extend(outliers(df, feature))
```

[625]:
```
index_list
```

[625]:
```
[84,
 567,
 734,
 795,
 1099,
 1451,
 1710,
 1739,
 1845,
 1989,
 2298,
 2876,
 2956,
 3225,
 3302,
 3349,
 3437,
```

3537,
3964,
3965,
4072,
4161,
4282,
4297,
4482,
4777,
4919,
5024,
5134,
5359,
5535,
5569,
5767,
5902,
6020,
6060,
6080,
6144,
6841,
6982,
7205,
7269,
7379,
7484,
7519,
7595,
7654,
7922,
7939,
7975,
8272,
8515,
8801,
8837,
8961,
9094,
9153,
9237,
9561,
9875,
10130,
10134,
10204,
10509,

10972,
11145,
11179,
11474,
11522,
11571,
12099,
12345,
12495,
12600,
12897,
12946,
13102,
13313,
13339,
13865,
14066,
14192,
14386,
14486,
14706,
14715,
15157,
15248,
15250,
15496,
15560,
15677,
15927,
16087,
16314,
16514,
16520,
16806,
16854,
16945,
17088,
17125,
17237,
17863,
17961,
18348,
18752,
18899,
19100,
19204,
19316,

19428,
19560,
19587,
19866,
20166,
20353,
20457,
20667,
20673,
20765,
20872,
20916,
20933,
21096,
21144,
21172,
21655,
21796,
21819,
21828,
22101,
22219,
22355,
22484,
22805,
22969,
23229,
23432,
23477,
23851,
23904,
24211,
24476,
24552,
25152,
25177,
25426,
25527,
25978,
26252,
26444,
26455,
26503,
26643,
26679,
27133,
27161,

27162,
27246,
27499,
27964,
28073,
28192,
28210,
28378,
28414,
28639,
28651,
28796,
28985,
29037,
29071,
29091,
29315,
29587,
29704,
30173,
30344,
30464,
30953,
31045,
31060,
31468,
31631,
31757,
31928,
31986,
31999,
32139,
32157,
32576,
32612,
32823,
33112,
33881,
33896,
34236,
34373,
34430,
34454,
34632,
34738,
34825,
34839,

35029,
35042,
35096,
35242,
35377,
35515,
35575,
35674,
35693,
36275,
36490,
36860,
36918,
37061,
37732,
37766,
37807,
38076,
38120,
38136,
38165,
38273,
38403,
38651,
39041,
39140,
39243,
39295,
39325,
39436,
39566,
39753,
39897,
40157,
40205,
40320,
40541,
40627,
40786,
40806,
40841,
40916,
41248,
41437,
41465,
41597,
41732,

41879,
41930,
42171,
42321,
42519,
42641,
42805,
43444,
43676,
44143,
44200,
44299,
44575,
44634,
44744,
44780,
44798,
45328,
45367,
45434,
45554,
45946,
45968,
46116,
46183,
46317,
46919,
47095,
47166,
47224,
47376,
47519,
47807,
47943,
48041,
48307,
48625,
48708,
48804,
49102,
49173,
49596,
49686,
49920,
49948,
50056,
50313,

50390,
50450,
50510,
50541,
50736,
50897,
50925,
50940,
51227,
51309,
51578,
51697,
51707,
51814,
51952,
52085,
52127,
52160,
52446,
52532,
52707,
52997,
53456,
54089,
54124,
54161,
54170,
54220,
54440,
54489,
54820,
55045,
55469,
55836,
55866,
55889,
56065,
56154,
56404,
56444,
56508,
56536,
56684,
56724,
56821,
57053,
57328,

57461,
57752,
57858,
57939,
58024,
58135,
58382,
58769,
58880,
59065,
59086,
59090,
59244,
59379,
59674,
59892,
59992,
60036,
60271,
60410,
60645,
60748,
61114,
61507,
61764,
61861,
62189,
62206,
62359,
62370,
62579,
62642,
62678,
62764,
62822,
63207,
63269,
63302,
63399,
63451,
63542,
63680,
63809,
63855,
63877,
64426,
64593,

64777,
64808,
64960,
65006,
65170,
65409,
65557,
65614,
66042,
66139,
66186,
66530,
66540,
66830,
67220,
67318,
67373,
67447,
67461,
67549,
67992,
68057,
68347,
68499,
68535,
68554,
68569,
68885,
69199,
69311,
69439,
69475,
69875,
70065,
70115,
70273,
70631,
70670,
70894,
71053,
71076,
71215,
71342,
71423,
71683,
71773,
71789,

71818,
71831,
72036,
72062,
72205,
72515,
72543,
72697,
72713,
72716,
72844,
72917,
72939,
73355,
73384,
73427,
73508,
73901,
73945,
74118,
74248,
74608,
75011,
75168,
76223,
76501,
76517,
76815,
76923,
77430,
77505,
77670,
78217,
78400,
78463,
79162,
79546,
79669,
79886,
80073,
80206,
80250,
80325,
80343,
80402,
80534,
80581,

80796,
80855,
80878,
80937,
80949,
80991,
81080,
81173,
81192,
81194,
81258,
81317,
81362,
81416,
81611,
81937,
82139,
82146,
82215,
82364,
82781,
82872,
83048,
83083,
83168,
83922,
84266,
84287,
84450,
84531,
84572,
84779,
84936,
85169,
85345,
85523,
85593,
85695,
85711,
85817,
85840,
85918,
85970,
86271,
86549,
86918,
87160,

87166,
87545,
87559,
87586,
87746,
87827,
87854,
87857,
87873,
87962,
88243,
88392,
88559,
88572,
88610,
88709,
88721,
88762,
89015,
89798,
89822,
90093,
90110,
90262,
90269,
90355,
90555,
90585,
90699,
90970,
91244,
91460,
92301,
92522,
92572,
92690,
92888,
93348,
93615,
93774,
93866,
94062,
94190,
94471,
94526,
94791,
94945,

95253,
95448,
95499,
96285,
96646,
96691,
96739,
96814,
96923,
97193,
97263,
97485,
97489,
97733,
97777,
97835,
97883,
97914,
97948,
98270,
99031,
99102,
99448,
99928,
100447,
100468,
100630,
100713,
100715,
100987,
101280,
101362,
101373,
101607,
102022,
102026,
102360,
102522,
102609,
102817,
102914,
103249,
103483,
104033,
104154,
104408,
104905,

105107,
105123,
105126,
105204,
105720,
105872,
105994,
105996,
106088,
106179,
106406,
106550,
106564,
106635,
106679,
107012,
107078,
107129,
107132,
107201,
107347,
107602,
108129,
108739,
109096,
109551,
109667,
109754,
109801,
109851,
109868,
109973,
109984,
110021,
110092,
110328,
110378,
110396,
110494,
110498,
110572,
110668,
110670,
110682,
110782,
110825,
111034,

111189,
111484,
111555,
111563,
111778,
112024,
112052,
112094,
112334,
112517,
112597,
113390,
113677,
113812,
113919,
114051,
114218,
114392,
114635,
114956,
114970,
114990,
115258,
115483,
115484,
115581,
115582,
115967,
116169,
116363,
116389,
116532,
116555,
116619,
116887,
116931,
116984,
117017,
117122,
117174,
117349,
117784,
118181,
118317,
118519,
118627,
118674,

118770,
118797,
119295,
119484,
119700,
119759,
120000,
120381,
120423,
120441,
120685,
121752,
121878,
122003,
122071,
122139,
122159,
122181,
122237,
122432,
122480,
122493,
122614,
122638,
122644,
122698,
122739,
123004,
123167,
123243,
123272,
123369,
123464,
123486,
123510,
123697,
123828,
124602,
124850,
125128,
125806,
126083,
126220,
126245,
126322,
126719,
126805,

126938,
126950,
126951,
126964,
126976,
127027,
127035,
127068,
127116,
127127,
127298,
127331,
127363,
127487,
127510,
127531,
127615,
128397,
128561,
128706,
128829,
129234,
129279,
129359,
129677,
129731,
129763,
129956,
129985,
130012,
130020,
130210,
130238,
130299,
130356,
130585,
130651,
130894,
130906,
130946,
131066,
131138,
131156,
131201,
131313,
131800,
131877,

131958,
132470,
132634,
132690,
132752,
132797,
132998,
133120,
133189,
133206,
133222,
133248,
133475,
133498,
133567,
133809,
133931,
134015,
134041,
134278,
134302,
134327,
134518,
134521,
134742,
135124,
135460,
135463,
135832,
135864,
136206,
136250,
136354,
136380,
136451,
136496,
136587,
136668,
136703,
136778,
136793,
136896,
137018,
137073,
137080,
137208,
137760,

137814,
138233,
138240,
138574,
138634,
138648,
138898,
139096,
139255,
139330,
139406,
139642,
139655,
139773,
140720,
141104,
141294,
141314,
141568,
141849,
142409,
142425,
142810,
142876,
143056,
143192,
143334,
143549,
143551,
143993,
144376,
144534,
144583,
144586,
144685,
145368,
145619,
145761,
145932,
146060,
146349,
146495,
146525,
146650,
146690,
146984,
147142,

147381,
147452,
147456,
147467,
147649,
147941,
147983,
148018,
148123,
148311,
148517,
148608,
148911,
148934,
149039,
149325,
149376,
149749,
149789,
149842,
150093,
150121,
150124,
150146,
150200,
150219,
150355,
150676,
150754,
151205,
151207,
151468,
151484,
151652,
151877,
151952,
152151,
152178,
152185,
152232,
152243,
152394,
152490,
152580,
152632,
152802,
152820,

```
153186,
153208,
153432,
153852,
153865,
153888,
153975,
154227,
154384,
154496,
154518,
154679,
154693,
154716,
154727,
154803,
154816,
154969,
155528,
155616,
156074,
156077,
156376,
156381,
156433,
156471,
156514,
156584,
156984,
157221,
157452,
157506,
157725,
157820,
157996,
158025,
158067,
158080,
158268,
158478,
158603,
159018,
159071,
…]
```

[626]:
```
# define a function called "remove" which returns a cleaned dataframe without␣
↪outliers
```

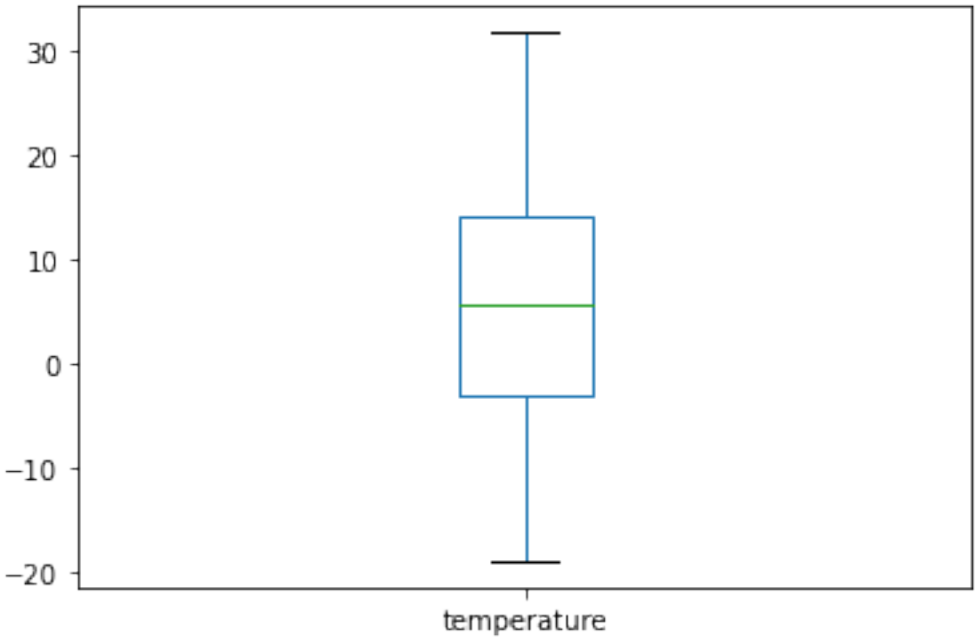
```
def remove(df,ls):
    ls=sorted(set(ls))
    df=df.drop(ls)
    return df
```

[627]:
```
df_cleaned=remove(df, index_list)
```

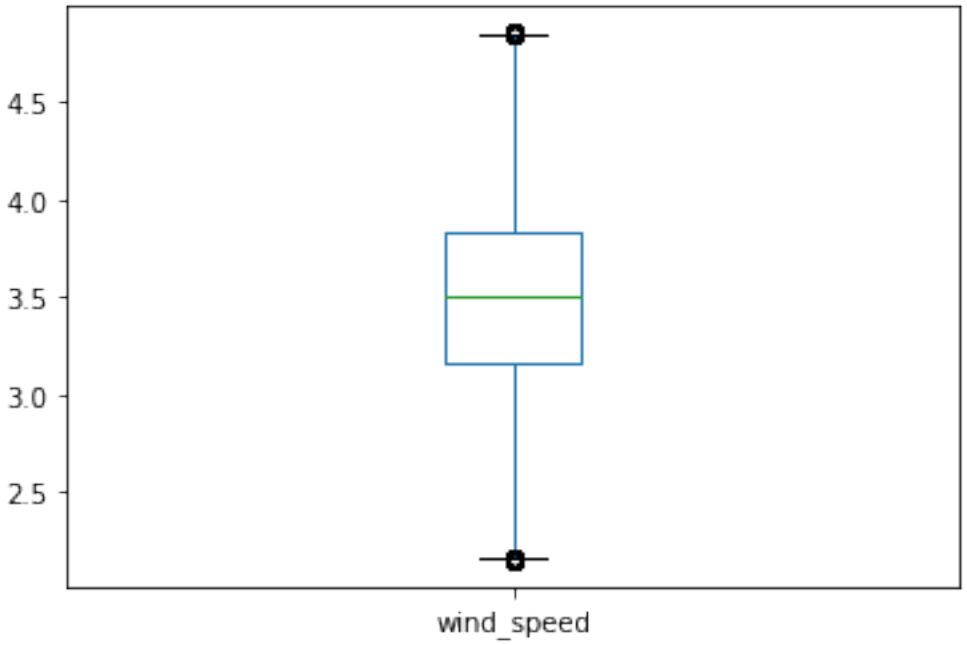
[628]:
```
df_cleaned.shape
```

[628]: (215570, 5)

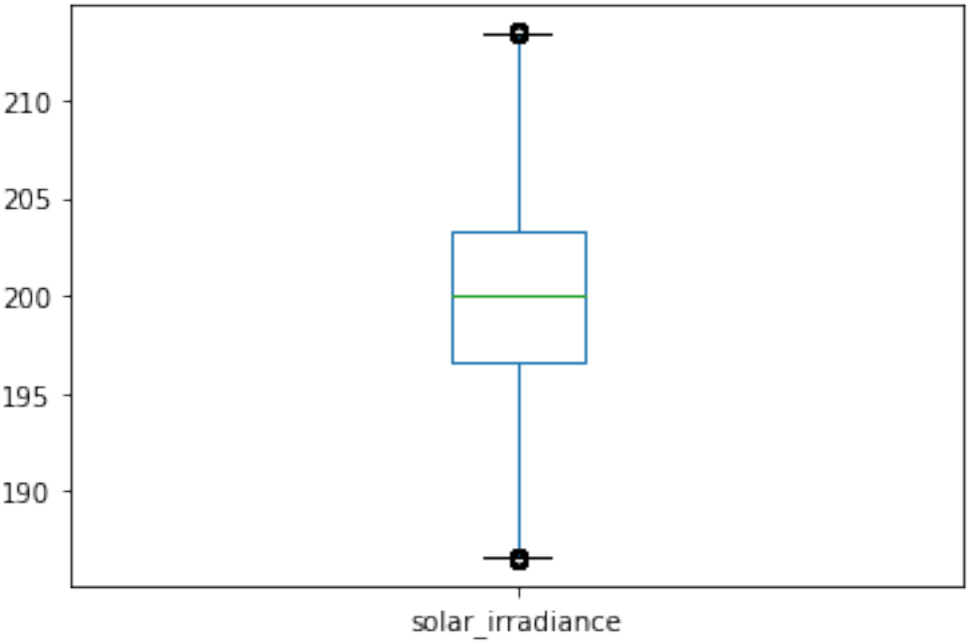
[629]:
```
plot_boxplot(df_cleaned, "temperature")
```



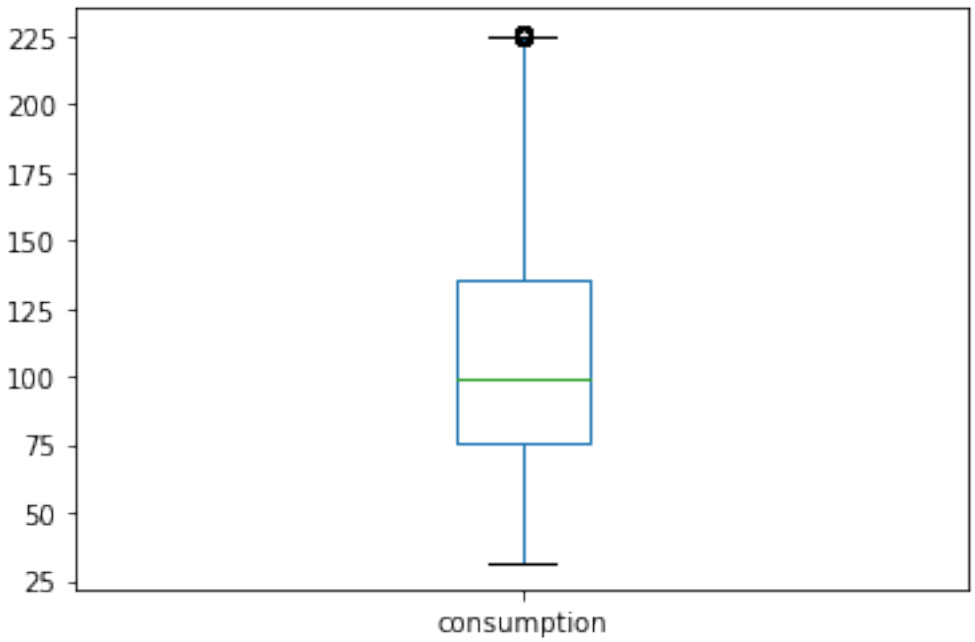
[630]:
```
plot_boxplot(df_cleaned, "wind_speed")
```



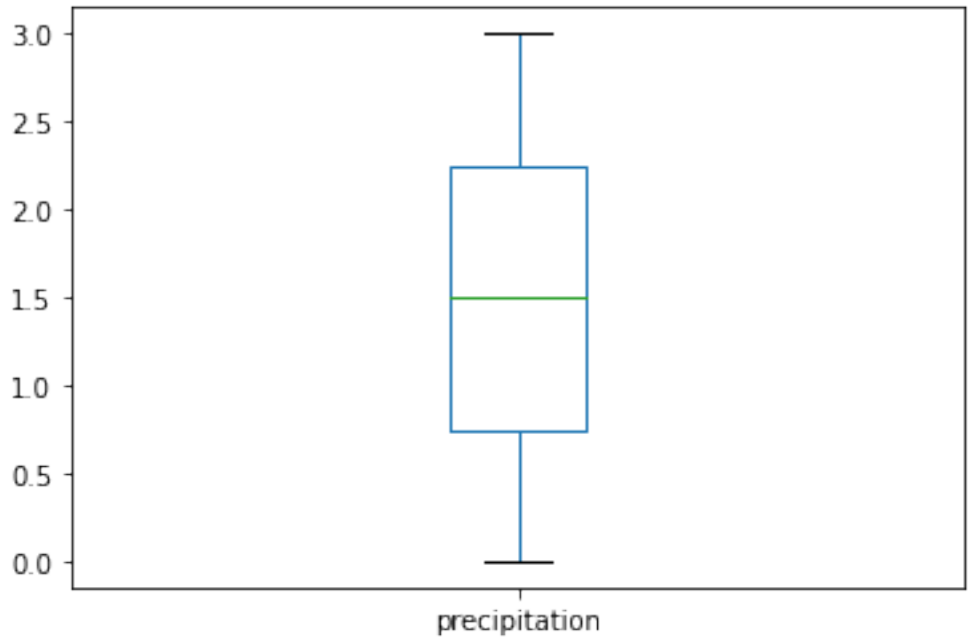
[631]:
```
plot_boxplot(df_cleaned, "solar_irradiance")
```



[632]:
```
plot_boxplot(df_cleaned, "consumption")
```



[633]:
```
plot_boxplot(df_cleaned, "precipitation")
```



[634]:
```
df_cleaned.to_csv( 'artificial_housing_data_cleaned.csv' , index=False)
```

[635]:
```
df=pd.read_csv ('artificial_housing_data_cleaned.csv')
df.head()
```

[635]:

	temperature	precipitation	wind_speed	solar_irradiance	consumption
0	12.856301	1.272970	4.708708	195.588130	77.184583
1	12.126253	2.814100	3.156671	203.112272	72.301535
2	15.277380	0.515568	3.490243	204.453061	68.666249
3	12.817692	1.920968	3.191047	192.645952	58.396602
4	14.343213	2.292260	3.142263	196.727594	76.230163

[636]:
```
df.describe
```

[636]:
```
<bound method NDFrame.describe of         temperature  precipitation  wind_speed
solar_irradiance  consumption
0         12.856301       1.272970    4.708708        195.588130    77.184583
1         12.126253       2.814100    3.156671        203.112272    72.301535
2         15.277380       0.515568    3.490243        204.453061    68.666249
3         12.817692       1.920968    3.191047        192.645952    58.396602
4         14.343213       2.292260    3.142263        196.727594    76.230163
…               …              …           …                …           …
215565     8.784783       1.499202    3.123499        197.511497   101.813038
215566     7.578867       0.003390    2.539654        194.567294    99.244315
215567     6.030384       2.695812    3.646082        193.759270    83.341239
215568     8.921952       0.987437    3.273055        193.294319    91.493841
215569     6.267010       2.824886    3.344415        197.644169    74.526816

[215570 rows x 5 columns]>
```

[637]:
```
11#separate the other attributes from the predicting attribute
X=df.drop(columns='consumption')
X
```

[637]:

	temperature	precipitation	wind_speed	solar_irradiance
0	12.856301	1.272970	4.708708	195.588130
1	12.126253	2.814100	3.156671	203.112272
2	15.277380	0.515568	3.490243	204.453061
3	12.817692	1.920968	3.191047	192.645952
4	14.343213	2.292260	3.142263	196.727594
…	…	…	…	…
215565	8.784783	1.499202	3.123499	197.511497
215566	7.578867	0.003390	2.539654	194.567294
215567	6.030384	2.695812	3.646082	193.759270
215568	8.921952	0.987437	3.273055	193.294319
215569	6.267010	2.824886	3.344415	197.644169

[215570 rows x 4 columns]

[638]:
```
12#separte the predicting attribute into Y for model training
y=df['consumption']
y
```

[638]:
```
0          77.184583
1          72.301535
2          68.666249
3          58.396602
4          76.230163
             …
215565    101.813038
215566     99.244315
215567     83.341239
215568     91.493841
215569     74.526816
Name: consumption, Length: 215570, dtype: float64
```

[639]:
```
13#Implement Ordinary Least Square
x=sm.add_constant(X)
results=sm.OLS(y,X).fit()
results.summary()
```

[639]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:            consumption   R-squared (uncentered):
0.892
Model:                            OLS   Adj. R-squared (uncentered):
0.892
Method:                 Least Squares   F-statistic:
4.434e+05
Date:                Wed, 29 Mar 2023   Prob (F-statistic):
0.00
Time:                        00:30:30   Log-Likelihood:
-1.0881e+06
No. Observations:              215570   AIC:
2.176e+06
Df Residuals:                  215566   BIC:
2.176e+06
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
====
                      coef    std err          t      P>|t|      [0.025
0.975]
```

```
------------------------------------------------------------------------------
----
temperature          1.1319      0.007    152.736      0.000       1.117
1.146
precipitation        0.2053      0.094      2.195      0.028       0.022
0.389
wind_speed           0.8409      0.164      5.119      0.000       0.519
1.163
solar_irradiance     0.4879      0.003    163.198      0.000       0.482
0.494
==============================================================================
Omnibus:                    20039.073   Durbin-Watson:                   0.202
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            15225.832
Skew:                           0.555   Prob(JB):                         0.00
Kurtosis:                       2.320   Cond. No.                         405.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1 importing train_test_split from sklearn

from sklearn.model_selection import train_test_split # splitting the data X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 10000)

[640]:
```
14# importing module
from sklearn.linear_model import LinearRegression
# creating an object of LinearRegression class
LR = LinearRegression()
```

[641]:
```
15# fitting the training data
LR.fit(X_train,y_train)
y_prediction =  LR.predict(X_test)
y_prediction
```

[641]:
```
array([121.08111239, 101.27047624,  88.64856025, …, 114.3980759 ,
       111.0341982 , 121.70550672])
```

[642]:
```
16# r2_score :Pearson correlation coefficient, ideal r2=1
from sklearn.metrics import r2_score
#Checking the R- squared value Or predicting the accuracy score
r_squared=r2_score(y_test,y_prediction)
```

```
r_squared
```

[642]: 0.09875283509840027

[643]:
```
17# Mean Square Error (MSE), ideal MSE=0
from sklearn.metrics import mean_squared_error
mean_sqrt_error=mean_squared_error(y_test,y_prediction)
mean_sqrt_error
```

[643]: 1398.4021374897372

[644]:
```
18#Root mean square error
import sympy
Root_mean_sqrt_error=sympy.sqrt(mean_sqrt_error)
Root_mean_sqrt_error
```

[644]: 37.3952154357979

[ ]: