**Tutorial da videoaula - Semana 7: Detecção de anomalias**

Neste exercício, usaremos métodos algorítmicos e estatísticos para detecção de anomalias. Começaremos com o algoritmo **Fator Local de Anomalia (LOF - Local Outlier Factor)** usando dados gerados com a **sklearn**. Em seguida, usaremos o diagrama de caixa e a análise de histograma, e vamos calcular os limites inferior e superior usando quartis em uma base de dados de moradias na Califórnia, disponível na biblioteca **sklearn**.

Este exercício é baseado no exemplo retirado do artigo Anomaly Detection Example with Local Outlier Factor in Python (<https://www.datatechnotes.com/2020/04/anomaly-detection-with-local-outlier-factor-in-python.html>) de Otabek Yorkinov, DataTechNotes.

1. Crie um novo notebook e inclua uma descrição para ele.

2. Nesta atividade, além da bibliotecas **pandas** e **numpy**, vamos importar as bibliotecas **sklearn** para usar o algoritmo LOF, a **matplotlib** para gerar gráficos e a **plotly** para fazer a análise de histograma. Importe as bibliotecas.

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.neighbors import LocalOutlierFactor

from sklearn.datasets import make\_blobs

plt.rcParams["figure.figsize"] = (15,10)

#### **Método algorítmico**

3. Gere os dados com o **make\_blobs**. Vamos definir os seguintes parâmetros:

* **número de amostras**: n\_samples=200
* **número de grupos**: centers=1
* **número de atributos**: n\_features=2
* **valores aleatórios**: random\_state=0, define o gerador valores aleatórios
* **área limite do agrupamento (x,y)**: center\_box=(10,10)
* **desvio padrão dos grupos**: cluster\_std=.3

Nesse caso, estamos gerando dados com um único grupo. O make\_blobs retorna o conjunto de dados.

dados, rotulos = make\_blobs(n\_samples=200, centers=1, n\_features=2, random\_state=0, center\_box=(10,10), cluster\_std=.3)

Alguns dos valores gerados:

dados[:10]

array([[10.01372756, 9.94384484],

[ 9.6188545 , 10.29081901],

[ 9.88453606, 10.02830548],

[ 9.84710435, 9.86857771],

[ 9.81157373, 9.85569186],

[ 9.96903434, 10.12317955],

[ 9.74163229, 10.57301949],

[10.01158917, 9.50298547],

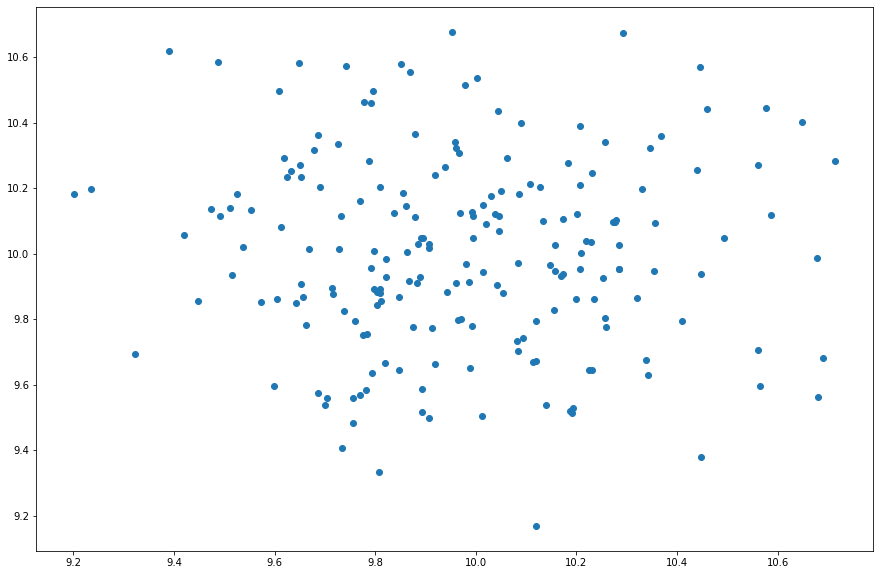
[10.23153717, 10.24705125],

[10.04648423, 10.11344876]])

4. Visualize os dados gerados usando o gráfico de dispersão.

plt.scatter(dados[:,0],dados[:,1])

plt.show()



5. Vamos usar o algoritmo Fator Local de Anomalia (LOF) para definir um modelo de predição, passando os k-vizinhos mais próximos e a contaminação, que define a proporção de anomalias no conjunto de dados.

k\_vizinhos = 20

contaminacao = .03

lof = LocalOutlierFactor(n\_neighbors=k\_vizinhos, contamination=contaminacao)

6. O passo seguinte é ajustar o modelo usando a função **fit\_predict()**

predicao = lof.fit\_predict(dados)

O resultado é uma lista na qual os valores negativos indicam as ocorrências de outliers.

Perceba que essa implementação do algoritmo é diferente do livro Introdução à Mineração de Dados quanto ao valor de saída das anomalias. Nas [equações](https://integrada.minhabiblioteca.com.br/reader/books/978-85-472-0100-5/pageid/311) apresentadas no livro, os valores muito maiores que 1 é que indicavam a ocorrência de anomalias. Mas, a lógica de identificação das anomalias é a mesma.

predicao

array([ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1,  
 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

7. Vamos visualizar as anomalias. Primeiro, vamos extrair somente as anomalias. A função **where()** retorna os índices que contêm o valor da comparação.

lof\_indice = np.where(predicao==-1)

anomalias = dados[lof\_indice]

8. Agora, vamos plotar todos os dados e, em seguida, as anomalias em vermelho. Vamos também rotular as anomalias para ter uma ideia de quais são os objetos que foram considerados como anomalias.

fig, eixo = plt.subplots()

eixo.scatter(dados[:,0], dados[:,1])

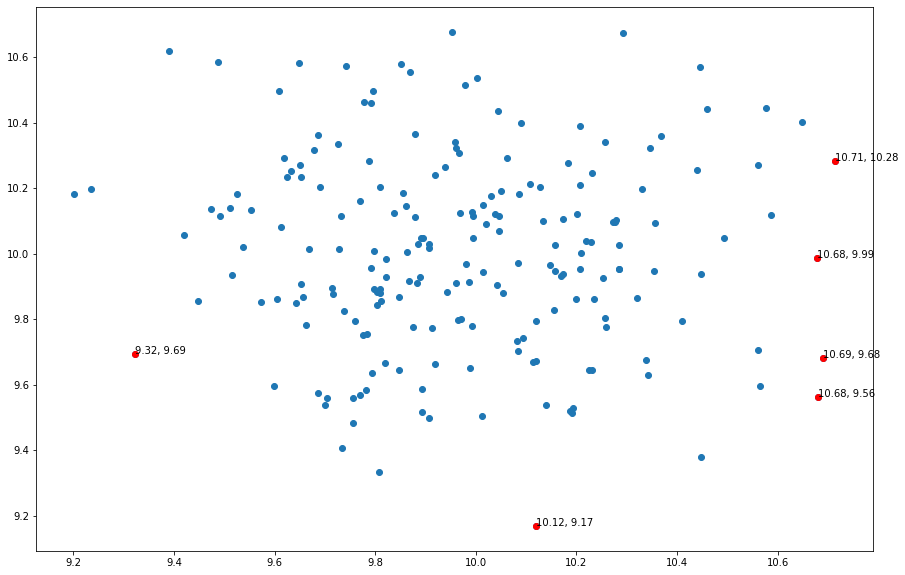
eixo.scatter(anomalias[:,0], anomalias[:,1], color='red')

#coloca os rótulos nos objetos anômalos (x, y)

for i, texto in enumerate(anomalias):

eixo.annotate('{:.2f}'.format(texto[0])+', '+'{:.2f}'.format(texto[1]), (anomalias[i,0], anomalias[i,1]))

plt.show()



Podemos ver que 6 objetos das extremidades inferior e direita do gráfico foram classificados como anomalias, que podem ser simplesmente identificados ou removidos de acordo com a necessidade.

**Métodos estatísticos**

Vamos fazer a detecção de anomalias usando métodos estatísticos como o diagrama de caixa, a análise de histograma e o cálculo do limite inferior e superior usando quartis.

Para esta análise, vamos usar um conjunto de dados sobre moradias na Califórnia e o **pandas** para visualizar os diagramas de caixa. A biblioteca **plotly** vai ser usada para visualizar também um exemplo de análise de histograma junto com o diagrama de caixa.

Este exercício é baseado no exemplo retirado do artigo 5 Easy Ways to Detect Outliers in Python (<https://datasciencesphere.com/analytics/5-easy-ways-to-detect-outliers-in-python/>) de Marc Bolle, Data Science Sphere.

9. Vamos importar e carregar a base de dados de moradias na Califórnia do censo de 1990 e a biblioteca **plotly**.

|  |
| --- |
| Essa base de dados tem os seguintes atributos:   * **MedInc**: Mediana de renda do quarteirão * **HouseAge**: Mediana de idade da residência * **AveRooms**: Média de ambientes por residência * **AveBedrms**: Média de quartos por residência * **Population**: Moradores do quarteirão * **AveOccup**: Média de moradores por residência * **Latitude** * **Longitude** |

from sklearn.datasets import fetch\_california\_housing

import plotly.express as pe

dados\_california = fetch\_california\_housing(as\_frame=True)

moradia = dados\_california.data

moradia.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 20640 entries, 0 to 20639

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MedInc 20640 non-null float64

1 HouseAge 20640 non-null float64

2 AveRooms 20640 non-null float64

3 AveBedrms 20640 non-null float64

4 Population 20640 non-null float64

5 AveOccup 20640 non-null float64

6 Latitude 20640 non-null float64

7 Longitude 20640 non-null float64

dtypes: float64(8)

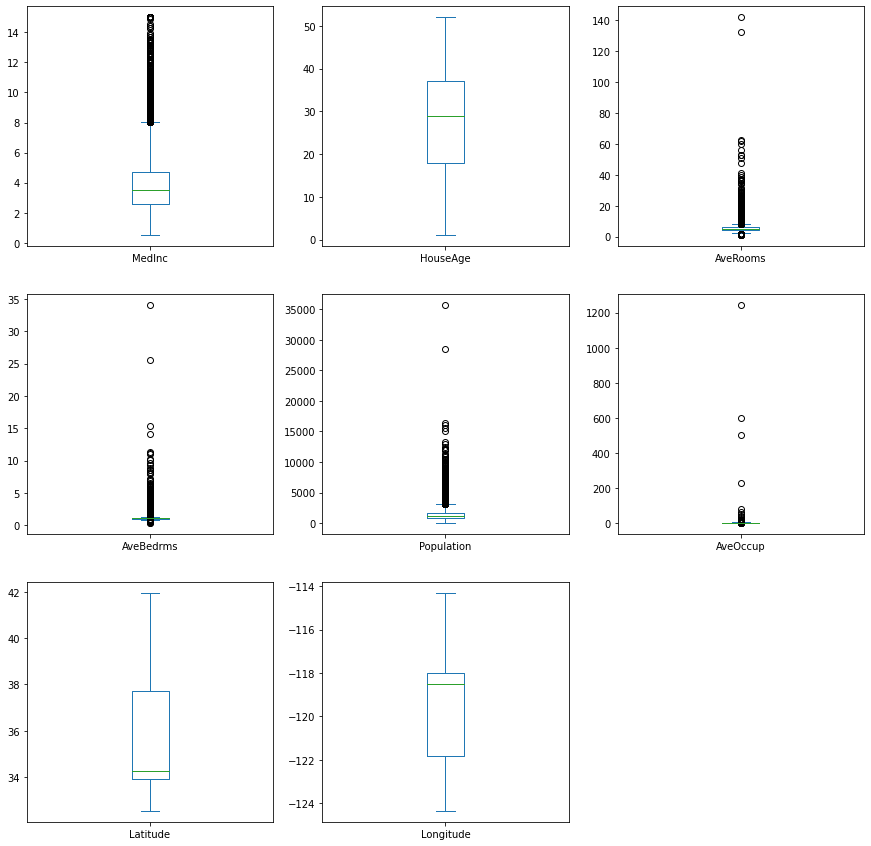
memory usage: 1.3 MB

10. Agora vamos plotar os boxplots de cada atributo, o que já vai dar uma ideia geral da distribuição desses dados e da ocorrência de anomalias. Alguns atributos como **MedInc** e **AveRooms** têm muitas anomalias, indicadas pelas circunferências acima ou abaixos dos quartis.

Definimos o tipo de gráfico (**kind='box'**), o uso de subgráficos (**subplots=True**), a quantidade de colunas e linhas (**layout=(3,3)**) e o tamanho da figura (**figsize=(15,15)**).

moradia.plot(kind='box', subplots=True, layout=(3,3), figsize=(15,15))

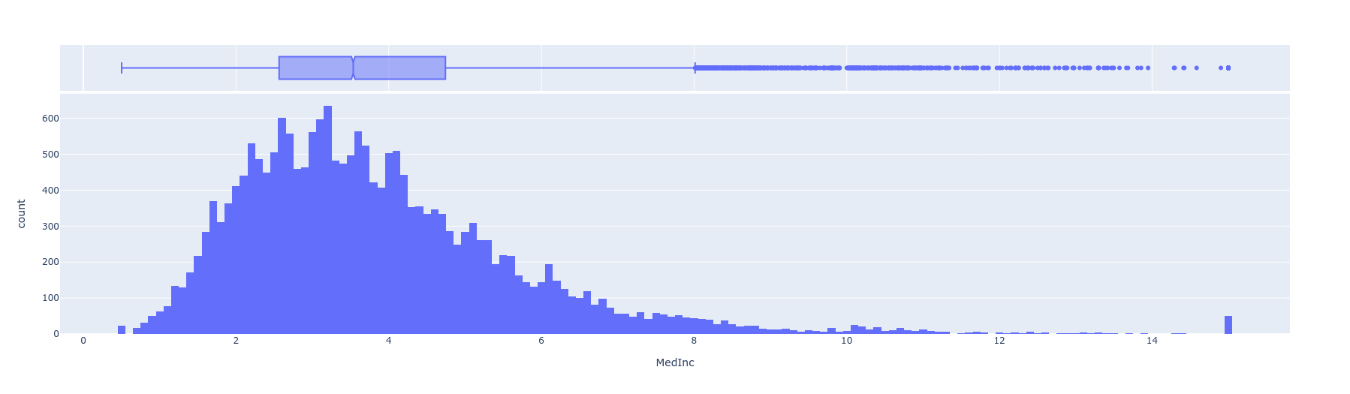
plt.show()



11. Vamos fazer também a análise de histograma de alguns desses atributos, começando pela mediana de renda (**MedInc**)

fig\_renda = pe.histogram(moradia, x='MedInc', marginal='box')

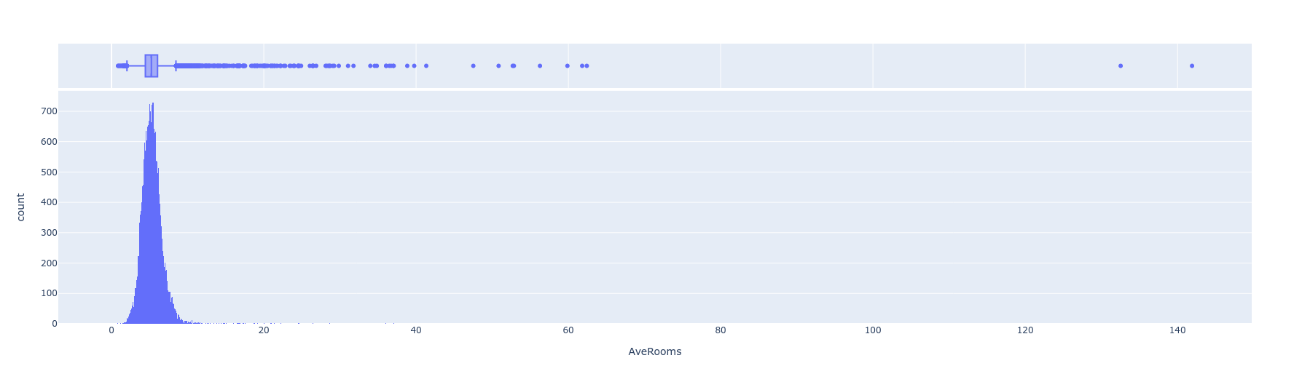
fig\_renda.show()



12. Ambientes (**AveRooms**)

fig\_ambientes = pe.histogram(moradia, x='AveRooms', marginal='box')

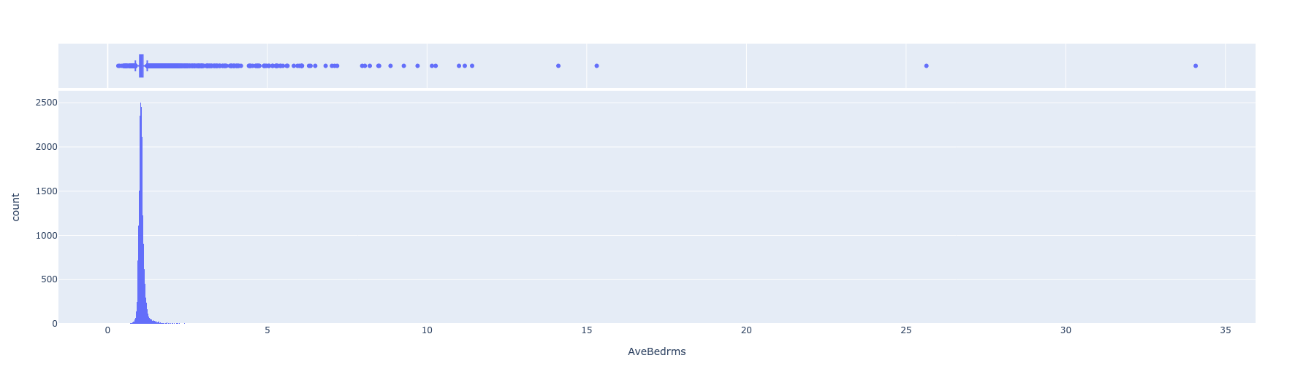
fig\_ambientes.show()



13. E quartos (**AveBedrms**)

fig\_quartos = pe.histogram(moradia, x='AveBedrms', marginal='box')

fig\_quartos.show()



14. Vamos visualizar esses dados em um diagrama de dispersão usando dois atributos com valores semelhantes, **AveRooms** e **AveBedrms**, média de ambientes e quartos, respectivamente. Podemos ver claramente anomalias no lado superior direito.

O parâmetro **alpha** adiciona uma transparência para os pontos, de forma que a sobreposição diminui a transparência, reforçando os locais de maior concentração de objetos.

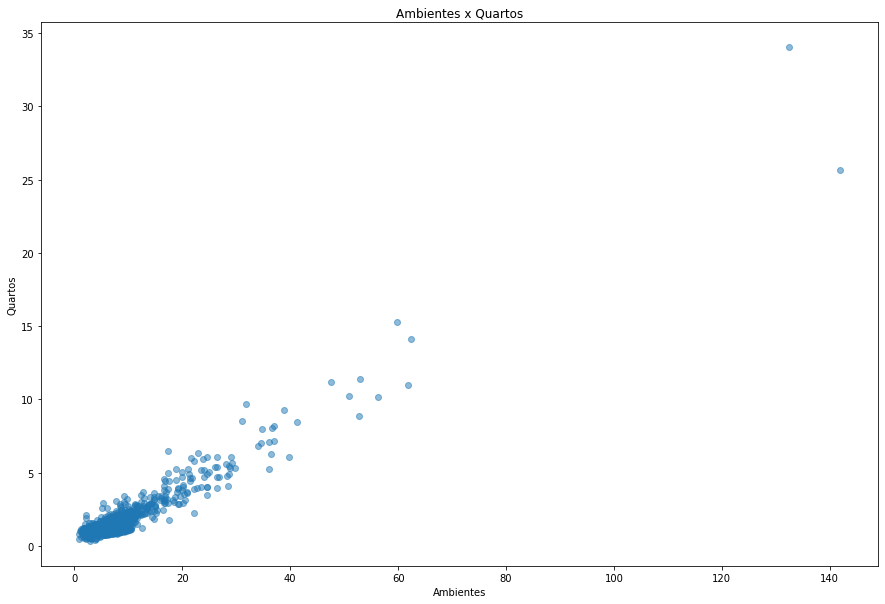
plt.scatter(moradia['AveRooms'], moradia['AveBedrms'], alpha=0.5)

plt.xlabel('Ambientes')

plt.ylabel('Quartos')

plt.title('Ambientes x Quartos')

plt.show()



15. Vamos também calcular os limites inferior e superior das anomalias calculando os quartis usados no diagrama de caixa no atributo população (**Population**).

populacao = moradia['Population']

populacao.head()

0 322.0  
1 2401.0  
2 496.0  
3 558.0  
4 565.0  
Name: Population, dtype: float64

16. Vamos calcular os valores do range interquartil, q1 e q3.

quartil1 = moradia['Population'].quantile(0.25)

quartil3 = moradia['Population'].quantile(0.75)

range\_interquartil = quartil3 - quartil1

print(f'Q1: {quartil1}, Q3: {quartil3} e RI: {range\_interquartil}')

Q1: 787.0, Q3: 1725.0 e RI: 938.0

17. Vamos definir os limiares inferior e superior usando o parâmetro **sigma = 1.5**

sigma = 1.5

limite\_inferior = quartil1 - (range\_interquartil \* sigma)

limite\_superior = quartil3 + (range\_interquartil \* sigma)

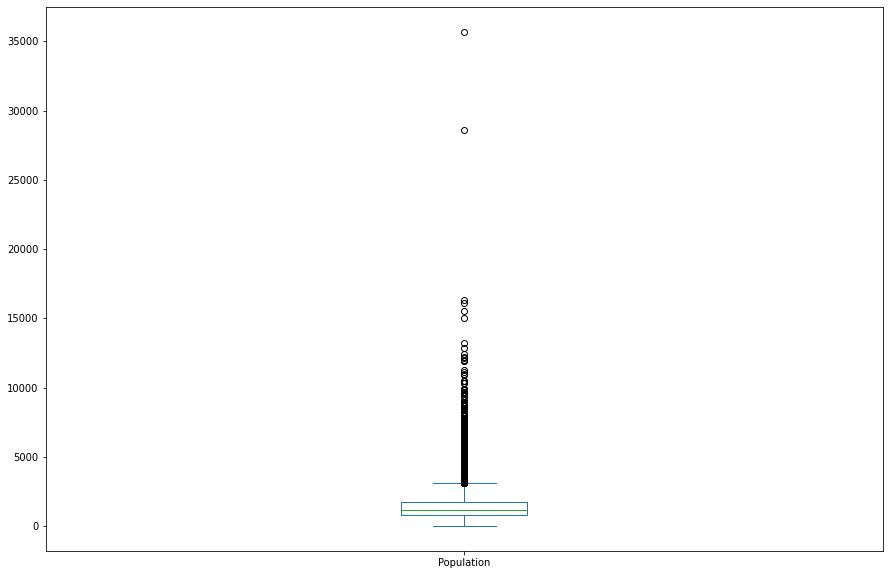
print(f'Limite inferior: {limite\_inferior}, limite superior: {limite\_superior}')

Limite inferior: -620.0, limite superior: 3132.0

18. Vamos plotar somente o atributo população para comparar os valores obtidos com o diagrama de caixa. Observando o diagrama de caixa, podemos ver que todos os valores acima do limite superior estão marcados como anomalias. Como o limite inferior é negativo, e não existem dados nessa faixa de valores, não há anomalias abaixo do limite inferior.

moradia['Population'].plot(kind='box')

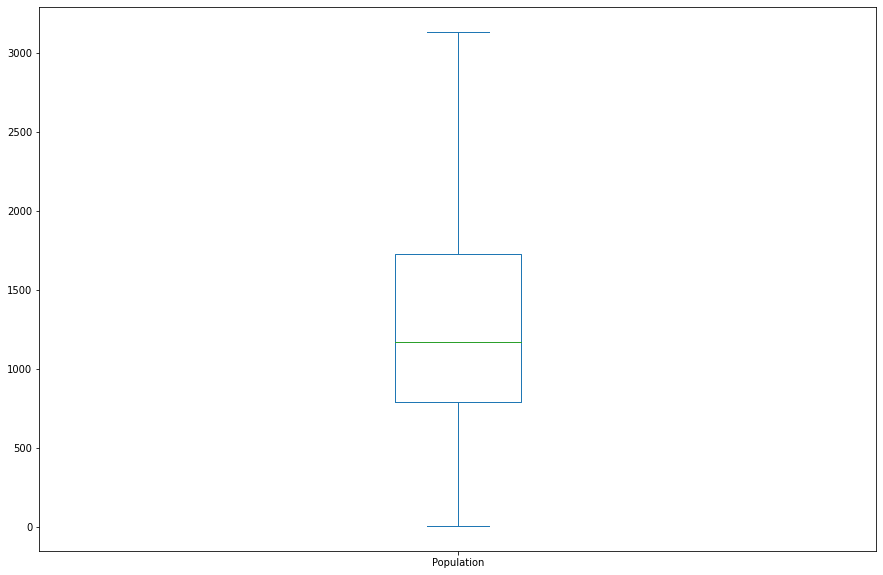
plt.show()



19. Vamos plotar removendo as anomalias, o que é feito através do parâmetro **showfliers=False**.

moradia['Population'].plot(kind='box', showfliers=False)

plt.show()



Com isso, concluímos esse tutorial. Analise outras bases de dados e veja os resultados. Tente usar também outros valores de parâmetros para ver as diferenças entre métodos estatísticos e algorítmicos, com um ou mais atributos etc.