

# Univariate analysis

## Two categorical and two numerical columns

```
In [1]: import pandas as pd
```

```
In [2]: data = pd.read_csv('star_dataset.csv') #from https://www.kaggle.com/datasets/deepu1109/s
```

```
In [3]: data.head()
```

```
Out[3]:
```

	Temperature (K)	Luminosity(L/L <sub>o</sub> )	Radius(R/R <sub>o</sub> )	Absolute magnitude(M <sub>v</sub> )	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	Red Dwarf	Red	M
1	3042	0.000500	0.1542	16.60	Red Dwarf	Red	M
2	2600	0.000300	0.1020	18.70	Red Dwarf	Red	M
3	2800	0.000200	0.1600	16.65	Red Dwarf	Red	M
4	1939	0.000138	0.1030	20.06	Red Dwarf	Red	M

## Categorical

### Star type

```
In [4]: data['Star type'].value_counts()
```

```
Out[4]: Red Dwarf      40
Brown Dwarf    40
White Dwarf    40
Main Sequence  40
Super Giant    40
Hyper Giant    40
Name: Star type, dtype: int64
```

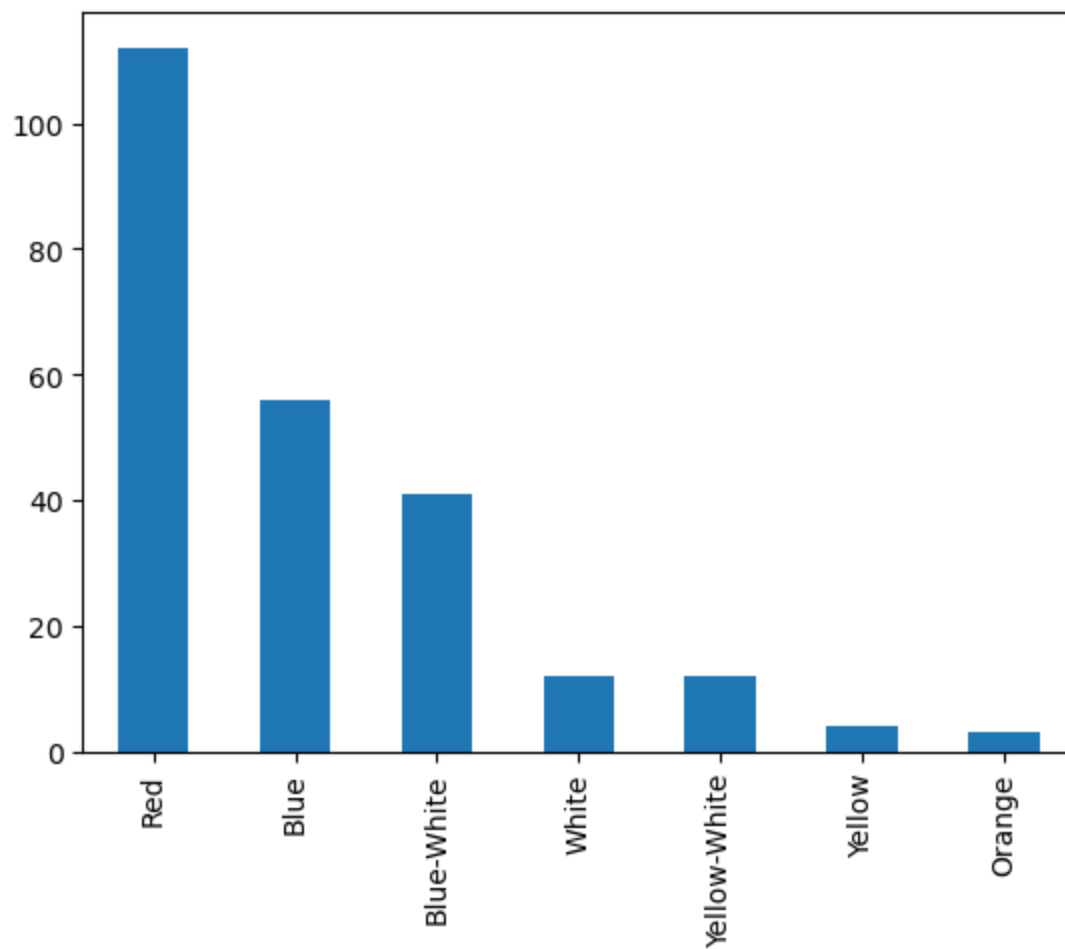
### Star color

```
In [5]: data['Star color'].value_counts()
```

```
Out[5]: Red      112
Blue      56
Blue-White  41
White     12
Yellow-White 12
Yellow     4
Orange     3
Name: Star color, dtype: int64
```

```
In [6]: data['Star color'].value_counts(dropna=False).plot(kind='bar')
```

```
Out[6]: <AxesSubplot:>
```



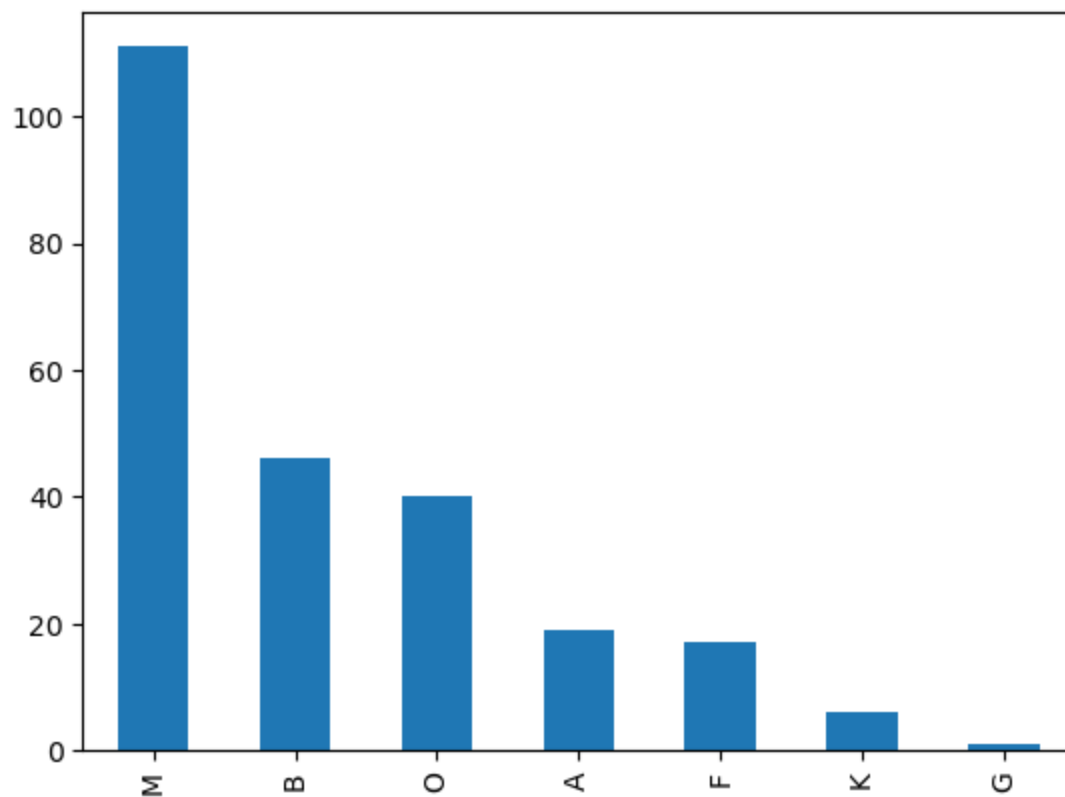
## Spectral class

```
In [7]: data['Spectral Class'].value_counts()
```







```
Out[7]: M    111  
       B     46  
       O     40  
       A     19  
       F     17  
       K      6  
       G      1  
       Name: Spectral Class, dtype: int64
```

```
In [8]: data['Spectral Class'].value_counts(dropna=False).plot(kind='bar')
```

```
Out[8]: <AxesSubplot:>
```



Er lijkt een correlatie te zijn tussen kleur en type, wat we ook zouden verwachten.

Main Sequence Stars						
						
Spectral Type:	O	B	A	F	G	K
Temperature:	40 000K	20 000K	8500K	6500K	5700K	4500K
Radius (Sun=1):	10	5	1.7	1.3	1.0	0.8
Mass (Sun=1):	50	10	2.0	1.5	1.0	0.7
Luminosity (Sun=1):	100 000	1000	20	4	1.0	0.2
Lifetime (million yrs):	10	100	1000	3000	10 000	50 000
Abundance:	0.00001%	0.1%	0.7%	2%	3.5%	8%

Giant Stars	White Dwarfs	Supergiant Stars
Low mass stars near the end of their lives.	Dying remnant of an imploded star.	High mass stars near the end of their lives.
Spectral Type: Mainly G, K or M	Spectral Type: D	Spectral Type: O, B, A, F, G, K or M
Temperature: 3000 to 10 000K	Temperature: Under 80 000K	Temperature: 4000 to 40 000K
Radius (Sun=1): 10 to 50	Radius (Sun=1): Under 0.01	Radius (Sun=1): 30 to 500
Mass (Sun=1): 1 to 5	Mass (Sun=1): Under 1.4	Mass (Sun=1): 10 to 70
Luminosity (Sun=1): 50 to 1000	Luminosity (Sun=1): Under 0.01	Luminosity (Sun=1): 30 000 to 1 000 000
Lifetime (million yrs): 1000	Lifetime (million yrs): -	Lifetime (million yrs): 10
Abundance: 0.4%	Abundance: 5%	Abundance: 0.0001%

Interessant is om nu nog te kijken of de waarden van de numerical kolommen ook overeenkomen met wat we verwachten.

```
In [9]: groupedByClass = data.groupby('Spectral Class').median()
```

```
groupedByClass.sort_values('Temperature (K)', ascending=False)
```

```
Out[9]:
```

	Temperature (K)	Luminosity(L/L <sub>o</sub> )	Radius(R/R <sub>o</sub> )	Absolute magnitude(M <sub>v</sub> )
--	-----------------	-------------------------------	---------------------------	-------------------------------------

Spectral Class				
----------------	--	--	--	--

<b>O</b>	22369.0	245865.00000	57.0000	-6.235
<b>B</b>	18850.0	0.03450	0.0146	10.365
<b>A</b>	9030.0	38.00000	2.4870	1.236
<b>F</b>	7230.0	0.00029	0.0130	12.020
<b>G</b>	6850.0	229000.00000	1467.0000	-10.070
<b>K</b>	4406.5	0.49350	1.0030	4.730
<b>M</b>	3324.0	0.00240	0.2910	13.120

Behalve de temperatuur, lijken de waarden niet helemaal te passen bij wat je zou verwachten van het bovenstaande. De reden hiervoor zie je onderin het bovenstaande plaatje, in een classificatie kunnen meerdere typen sterren zitten, met een vergelijkbare temperatuur maar een hele andere radius en daarmee ook luminosity.

Wanneer je de waarden tegenover het type zet, zie je dat de luminosity en radius wel relateren, maar temperatuur geen verband vertoont.

```
In [10]: groupedByType = data.groupby('Star type').median()  
groupedByType.sort_values('Radius (R/Ro)', ascending=False)
```

```
Out[10]:
```

	Temperature (K)	Luminosity(L/L <sub>o</sub> )	Radius(R/R <sub>o</sub> )	Absolute magnitude(M <sub>v</sub> )
--	-----------------	-------------------------------	---------------------------	-------------------------------------

Star type				
-----------	--	--	--	--

<b>Hyper Giant</b>	3766.0	266500.00000	1352.5000	-9.915
<b>Super Giant</b>	12821.0	242145.00000	43.5000	-6.235
<b>Main Sequence</b>	12560.5	738.50000	5.7125	-1.180
<b>Brown Dwarf</b>	3314.0	0.00315	0.3380	12.605
<b>Red Dwarf</b>	2935.0	0.00052	0.1060	17.145
<b>White Dwarf</b>	13380.0	0.00076	0.0102	12.340

## Numerical

### Radius

```
In [11]: radius = 'Radius (R/Ro)'  
rFilter = data[radius]
```

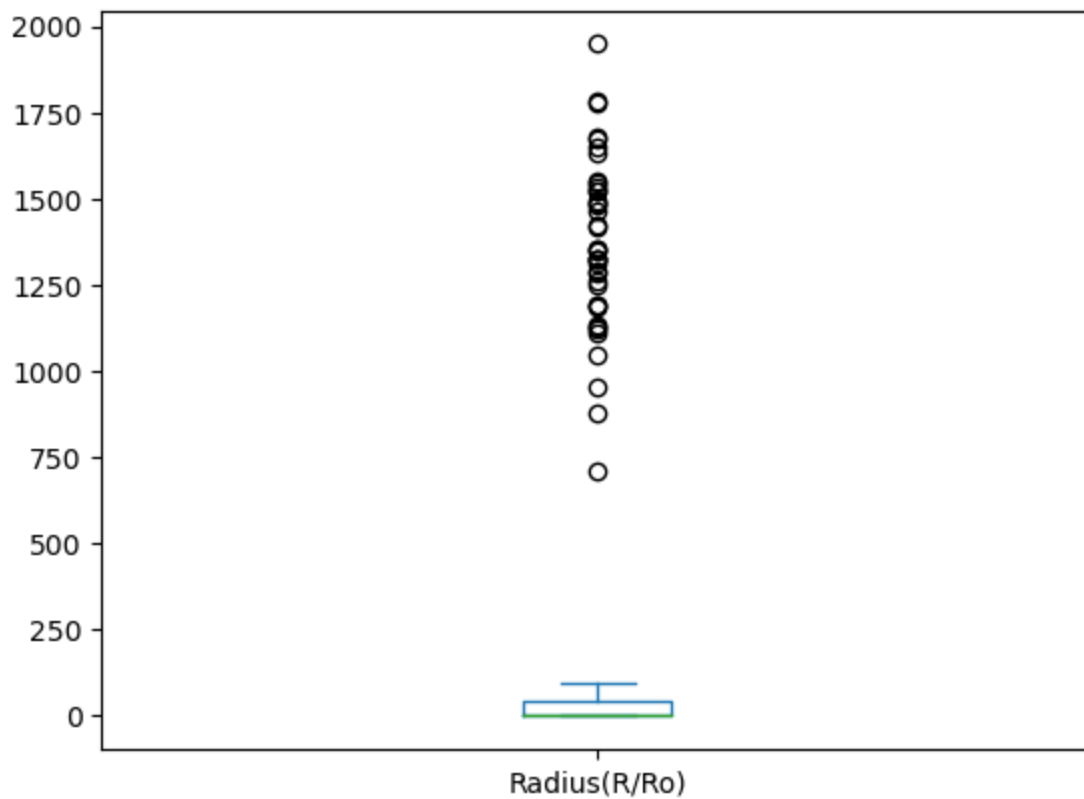
```
In [12]: print("Minimum: " + str(rFilter.min()))  
print("Maximum: " + str(rFilter.max()))  
print("Mean: " + str(rFilter.mean()))  
print("Median: " + str(rFilter.median()))  
print("Standard deviation: " + str(rFilter.std()))
```

```
Minimum: 0.0084  
Maximum: 1948.5  
Mean: 237.15778137500004
```

Median: 0.7625  
Standard deviation: 517.1557634028478

```
In [13]: rFilter.plot(kind='box')
```

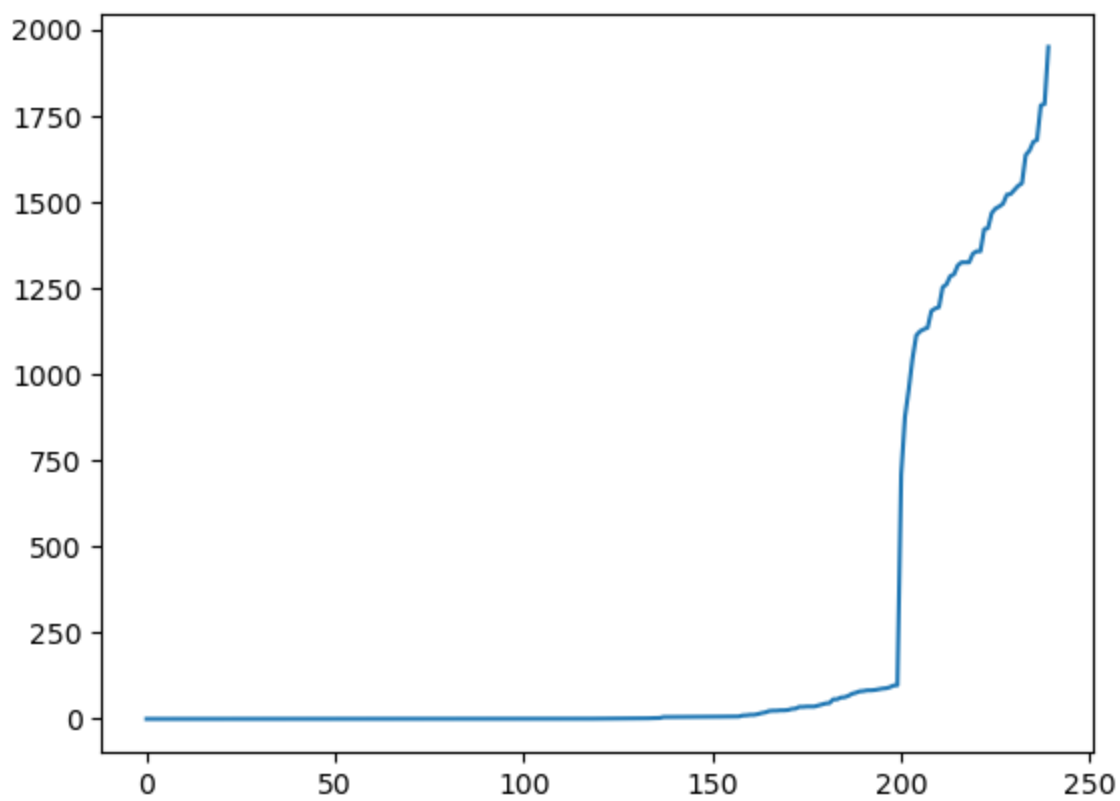
```
Out[13]: <AxesSubplot:>
```



Er zit hier een vrij groot verschil tussen de mediaan en het gemiddelde. Om een beter beeld te krijgen van de meeste waarden in de dataset kunnen we de outliers uitfilteren, echter doordat de dataset dan steeds minder grote waarden heeft, verplaatst de boxplot steeds verder naar beneden. Ook kan je in de volgende plot zien dat, hoewel er een grote sprong is rond de 100, er geen losstaande waarden zijn die heel erg afwijken, ze omvatten ongeveer 1/5 van de gehele dataset. Dat ze niet in de boxplot worden meegenomen is logisch, omdat die de eerste 3/4 van de dataset beschrijft, waar de laatste kwart is gebaseerd op de afstand van de mediaan tot het getal wat op 3/4 ligt.

```
In [14]: rFilter.sort_values(ignore_index=True).plot()
```

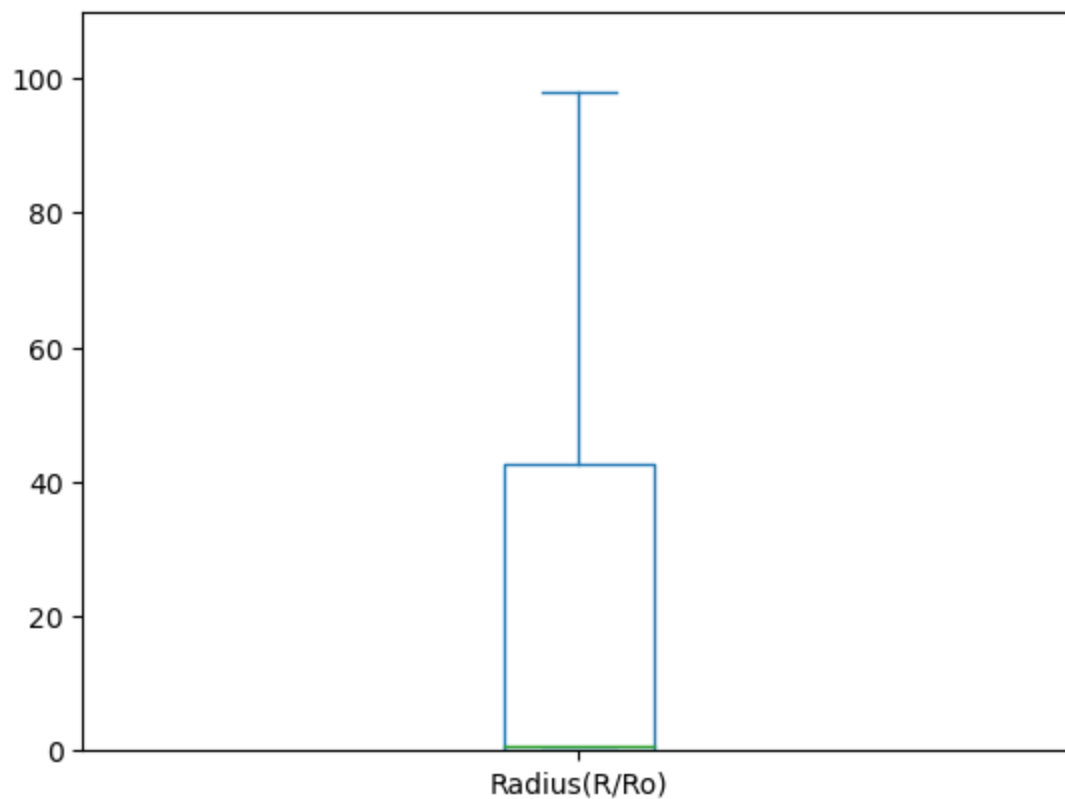
```
Out[14]: <AxesSubplot:>
```



Voor een iets duidelijker beeld kunnen we wel de y verplaatsen naar de sprong die we hierboven zien.

```
In [15]: rFilter.plot(kind='box', ylim=[0, 110])
```

```
Out[15]: <AxesSubplot:>
```



## Temperature

```
In [16]: temperature = 'Temperature (K)'
tFilter = data[temperature]
```

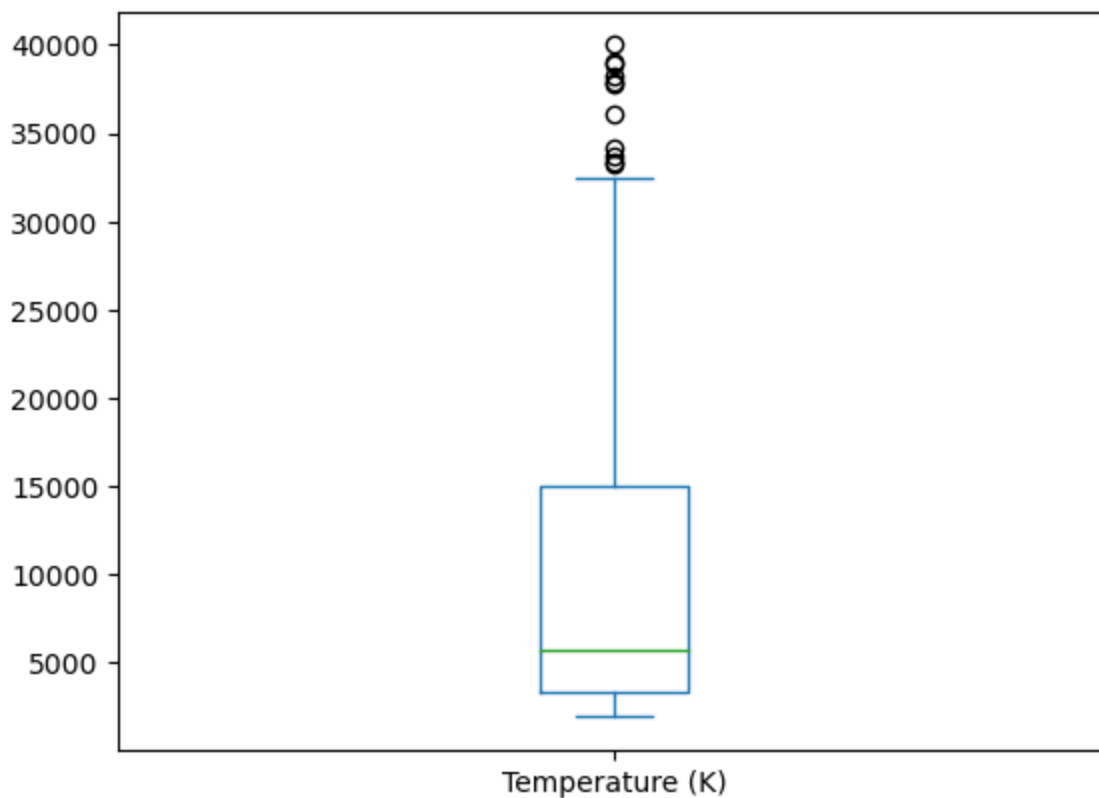
```
In [17]: print("Minimum: " + str(tFilter.min()))
print("Maximum: " + str(tFilter.max()))
print("Mean: " + str(tFilter.mean()))
print('Median: ' + str(tFilter.median()))
print('Standard deviation: ' + str(tFilter.std()))
```

```
Minimum: 1939
Maximum: 40000
Mean: 10497.4625
Median: 5776.0
Standard deviation: 9552.42503716402
```

Ook hier is het gemiddelde een stuk hoger dan de mediaan. Daarnaast kunnen we afleiden dat de temperatuur niet met een rechte lijn gefit kan worden, omdat de standard deviation (gebaseerd op afstand tot het gemiddelde) erg groot is.

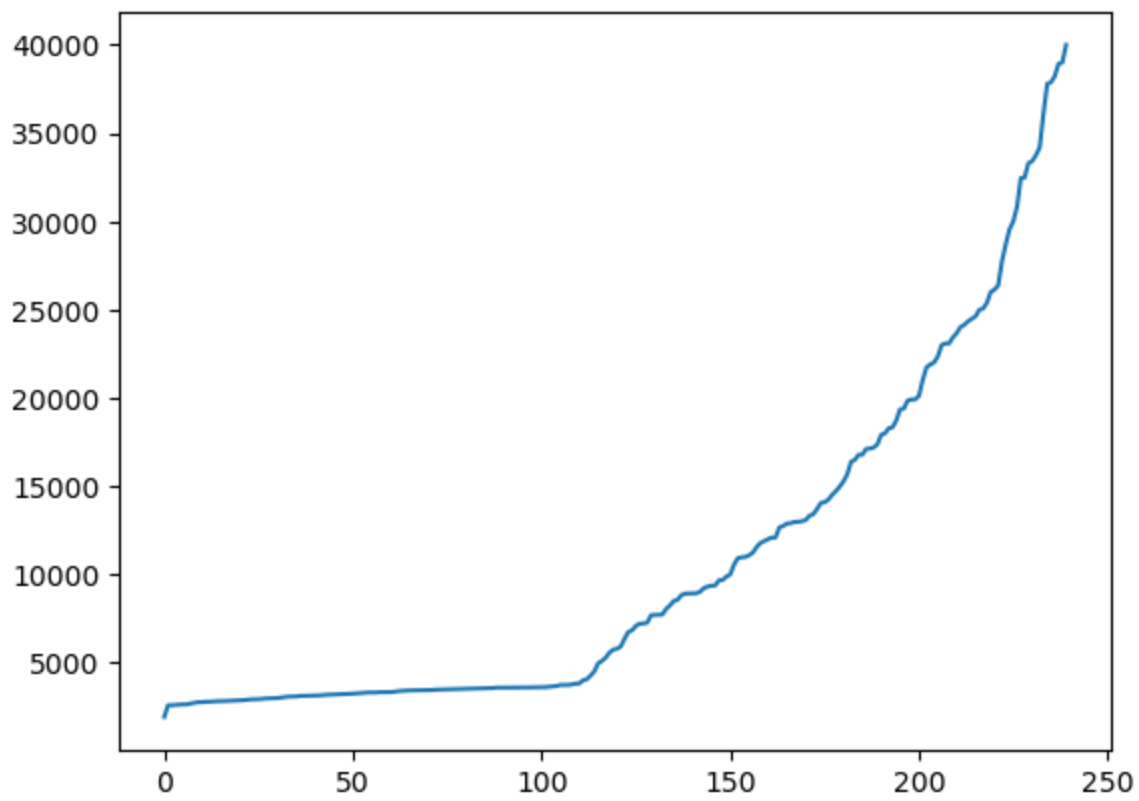
```
In [18]: tFilter.plot(kind='box')
```

```
Out[18]: <AxesSubplot:>
```



```
In [19]: tFilter.sort_values(ignore_index=True).plot()
```

```
Out[19]: <AxesSubplot:>
```



Ook hier vind je waarden die buiten het maximum van de boxplot liggen, maar in deze dataset kunnen deze waarden moeilijk als outliers gezien worden.