From UCI Machine Learning Repository

In [ ]:
```
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
plt.style.use('ggplot')
```

Select the dataset

Read data from archive.

Look at the data and try to understand if:

1. if it is a `csv` file or other
2. for `csv`, what is the *separator* character (`,`, `;`, `\t`, ...)
3. for `csv`, is there a *header*? it is a first row containing column names
4. if there is no header, look for reasonable names, e.g. for *UCI* a `.names` file
5. if there is no header, look at the documentation of `read_csv` to see how to specify column names
6. try to understand if the dataset is supervised, and what is the *target Class*

The download url is https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Use the read_csv() method of pandas dataframe https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Use `df` as the dataframe name

Assign column names if necessary

In [ ]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
# url= './iris.csv'
# adjust the line below, if necessary
# df = pd.read_csv(url, names=["Sepal Length", "Sepal Width", "Petal Length", "Peta

url = './adult.data'
df = pd.read_csv(url, names = ['age', 'workclass', 'fnlwgt', 'education', 'educati
```

Show column names

Use the `columns` attribute of pandas on `df`

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'Class'],
      dtype='object')
```

Show portion of data

Use the `head` method of pandas dataframe

In [ ]:
```
df.head()
```

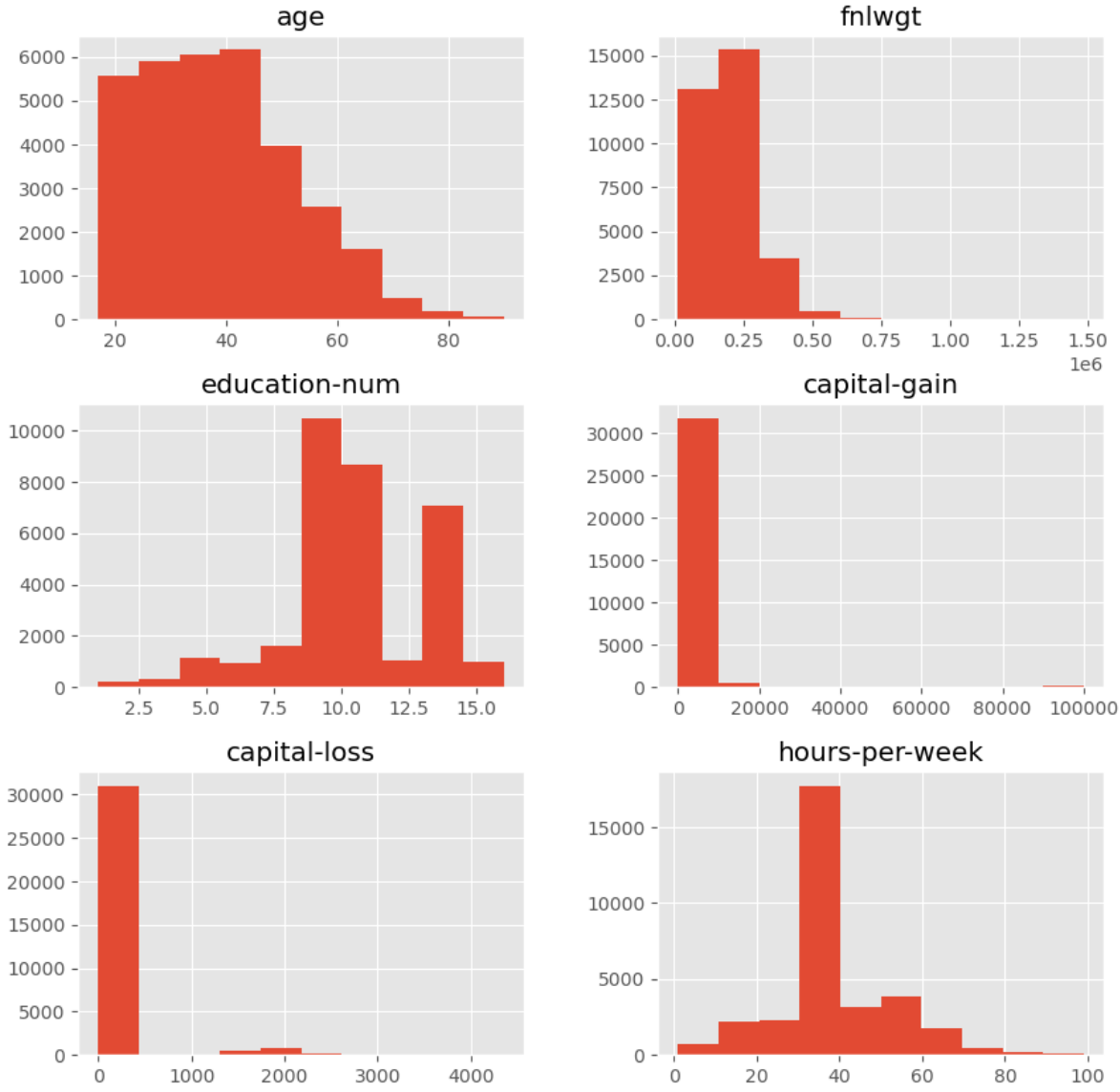
Out[ ]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fen

Show histograms for all numeric values

Use the `DataFrame.hist` method of Pandas. You can set the `figsize` parameter to adjust size

In [ ]:
```
pd.DataFrame.hist(df
                  , figsize = [10,10]
                 );
```



Is there anything to observe? balanced distributions? skewed distributions? outliers?

Show synthetic description

The **describe** method of pandas dataframes gives a short summary

Examine in the documentation if there are interesting options in the method

In [ ]:
```
df.describe()
```

Out[ ]:

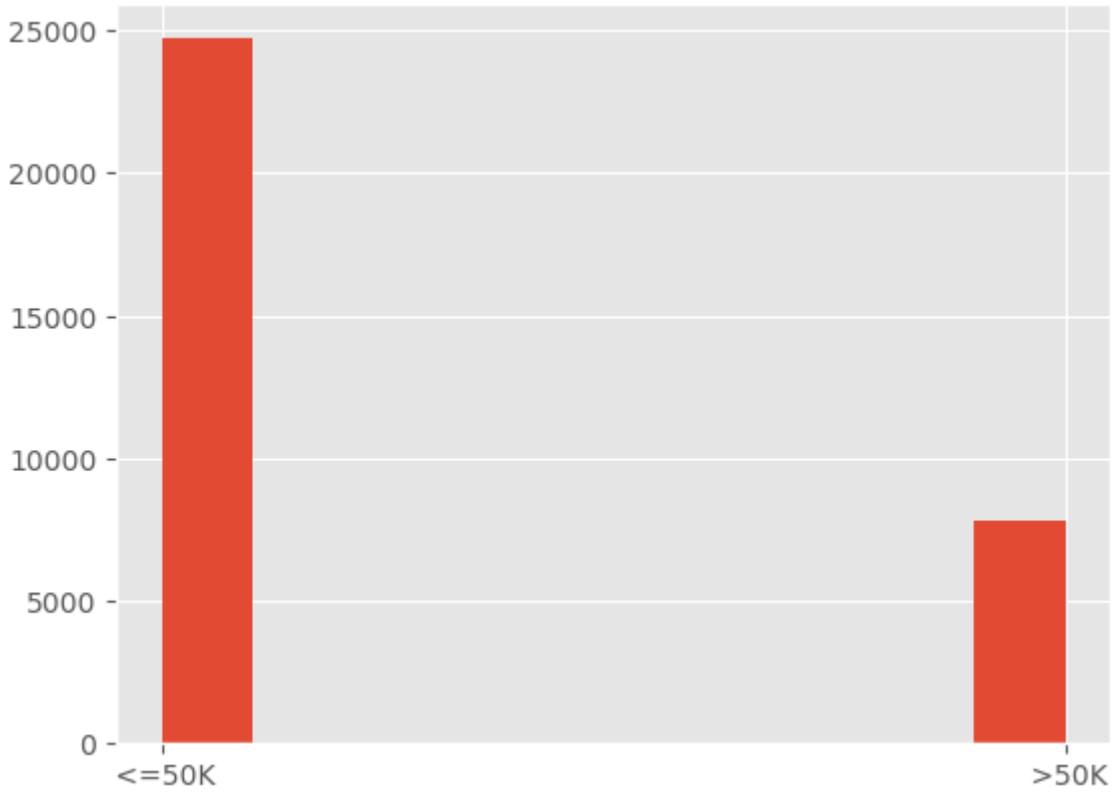
	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Are there *missing values*? How could we see it from the description?

Plot an histogram for "the target column"

Use the `hist` method of `matplotlib.pyplot` applied to the target column of `df`

In [ ]:
```
### adjust the line below
plt.hist(df['Class'])
plt.show()
```



Pairplot

The `pairplot` of the *Seaborn* library is a powerful data exploration tool. It shows a plot of pairs of numeric attributes, and may represent as color the attribute chosen as Class (the

`hue` parameter). In this specific case the high number of attributes makes the representation not very clear.

Use df as argument to the `pairplot` method of Seaborn, specifying also `hue = '...'` and `diag_kind='kde'` (try also other options)

In [ ]:
```
sns.pairplot(df, hue = 'Class'
             , diag_kind='kde'
            );
```

Show the *correlation*

Documentation Wikipedia Reference

In [ ]:
```
corr = df[df.columns].corr()
plt.figure(figsize=(15,10)) # set X and Y size
sns.heatmap(corr, cmap="YlGnBu", annot=True);
```

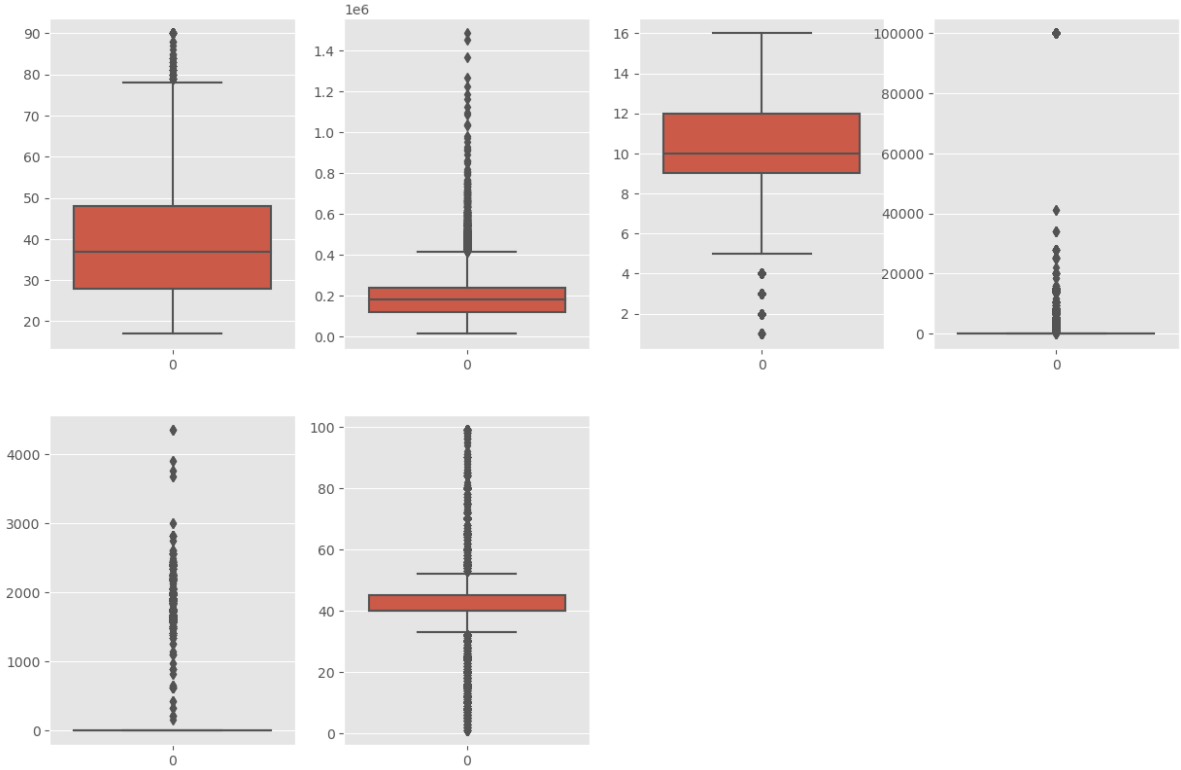
Boxplot

Now we will explore the distribution of the values inside each column using the `boxplot`.

This kind of plot shows the three quartile values of the distribution along with extreme values. The "whiskers" extend to points that lie within 1.5 IQRs of the lower and upper quartile, and then observations that fall outside this range are displayed independently. This means that each value in the boxplot corresponds to an actual observation in the data (*from the official Seaborn documentation*)

In [ ]:
```
#--Checking Outliers
plt.figure(figsize=(15,15))
pos = 1
true_columns = df.columns[df.dtypes == "int64"]
print(true_columns)
for i in true_columns:
    plt.subplot(3, 4, pos)
    sns.boxplot(data=df[i])
    pos += 1
```

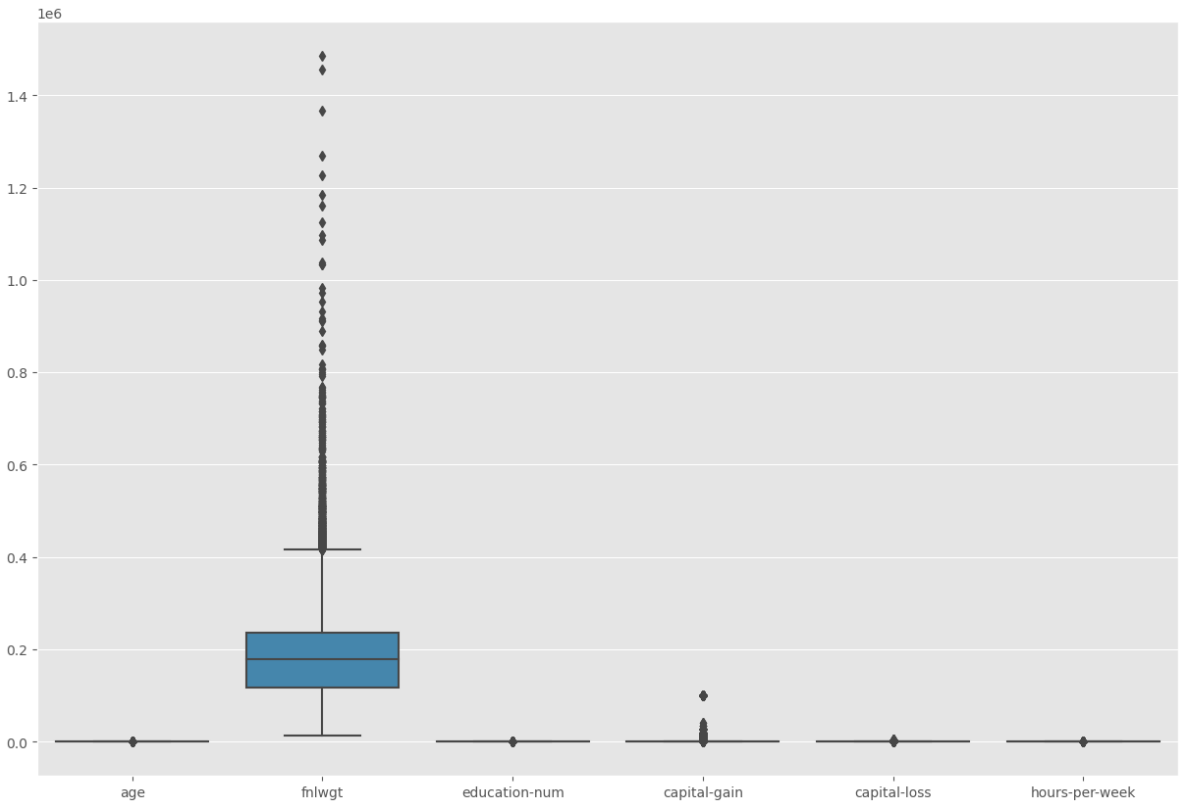
```
Index(['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
       'hours-per-week'],
      dtype='object')
```



Comment what you see, are there relevant situations? outliers?

Another way to produce a *boxplot*

In [ ]:
```
plt.figure(figsize=(15,10))
sns.boxplot(data = df);
```

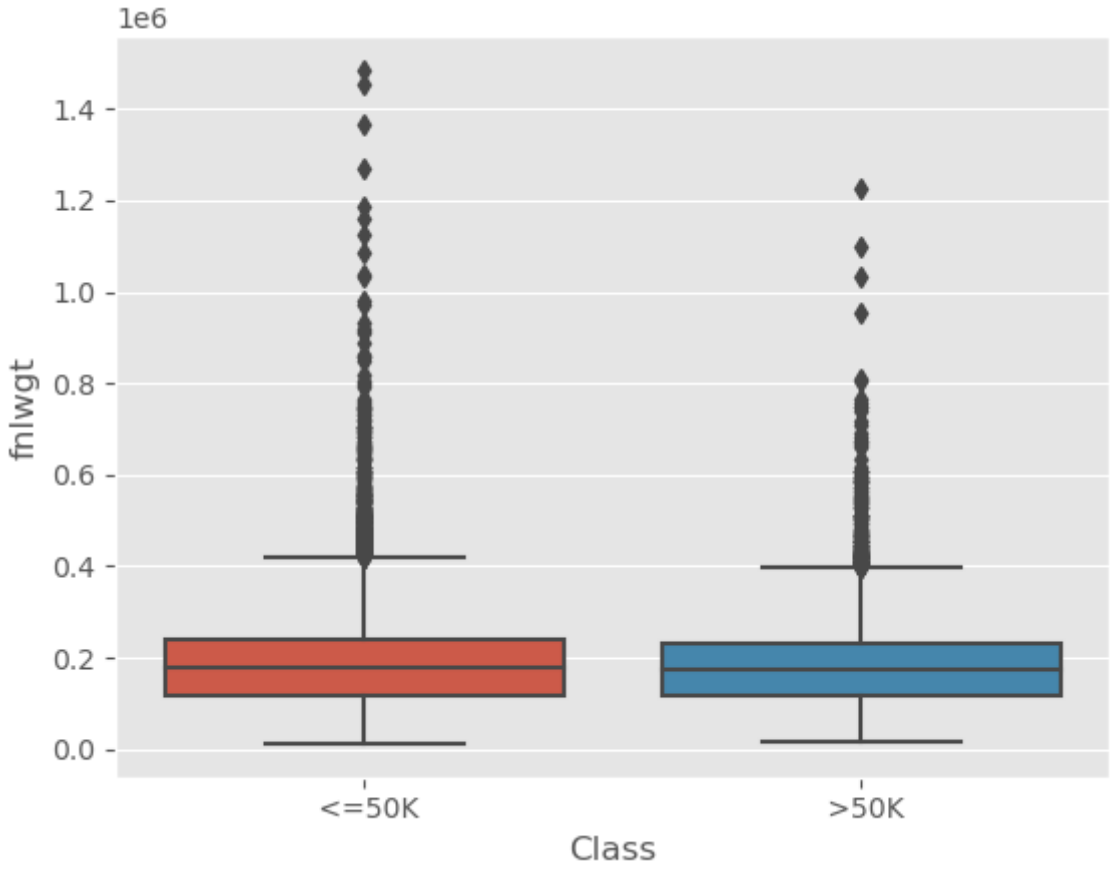


A boxplot for an attribute and the target

Put the attribute in the `y` axis, the target in the `x` axis

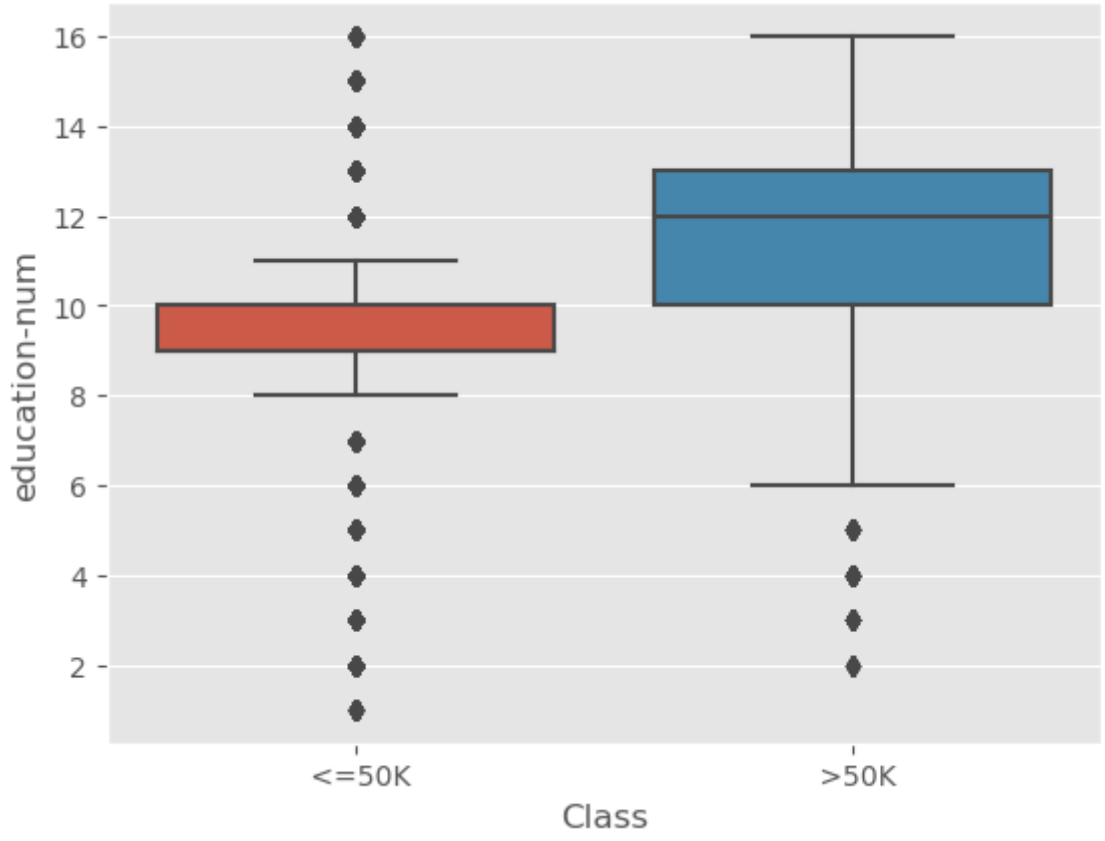
In [ ]:
```
sns.boxplot(x='Class', y='fnlwgt', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='fnlwgt'>
```



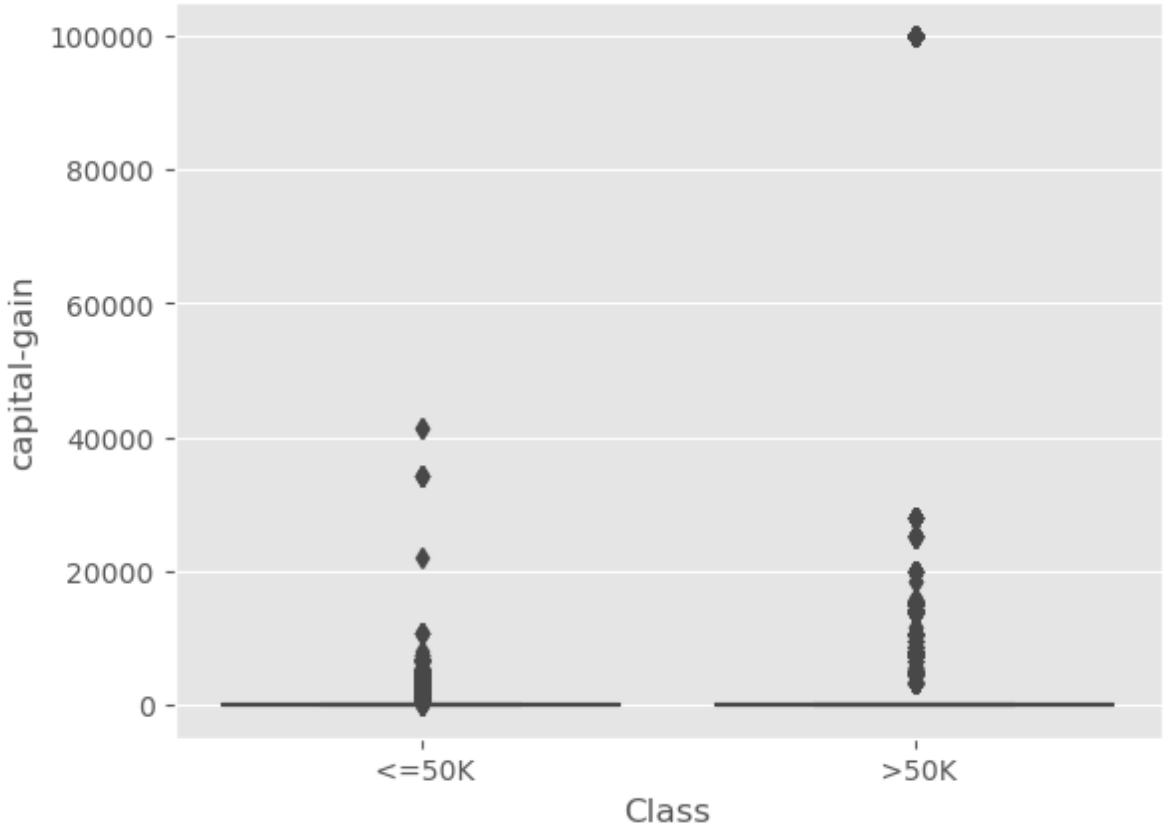
In [ ]:
```
sns.boxplot(x='Class', y='education-num', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='education-num'>
```



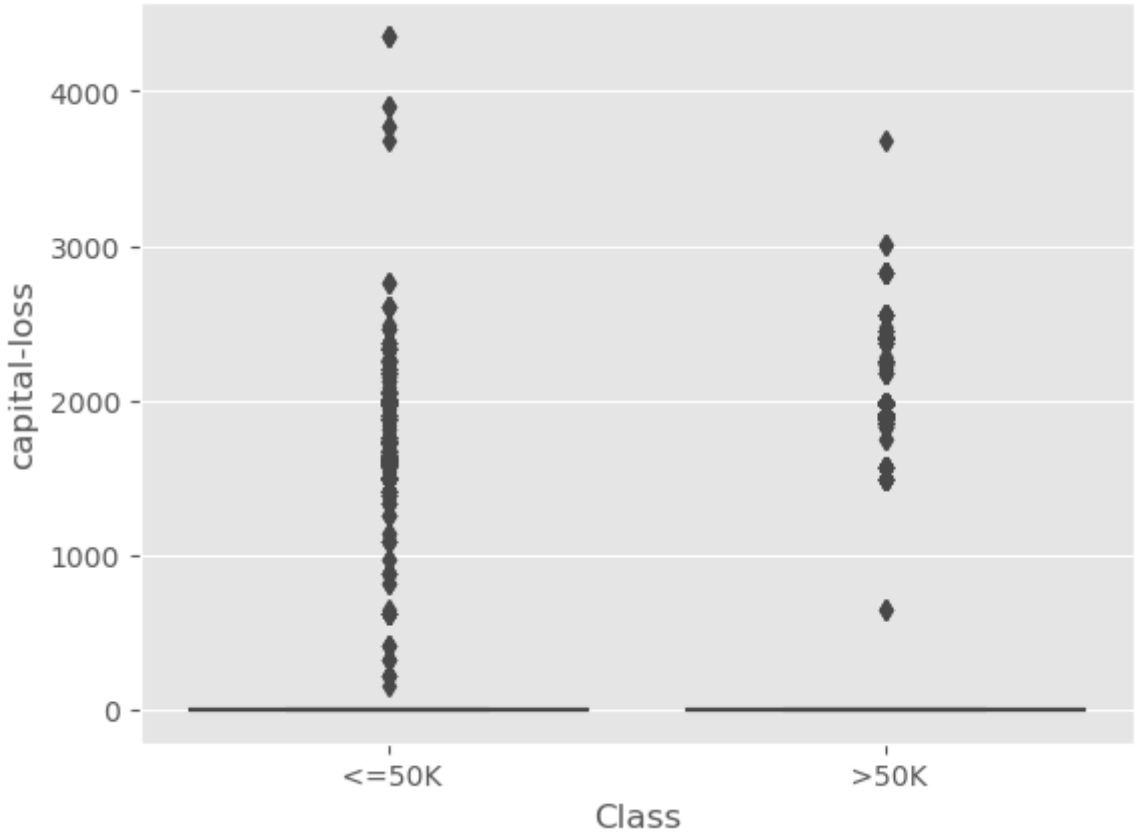
In [ ]:
```
sns.boxplot(x='Class', y='capital-gain', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='capital-gain'>
```



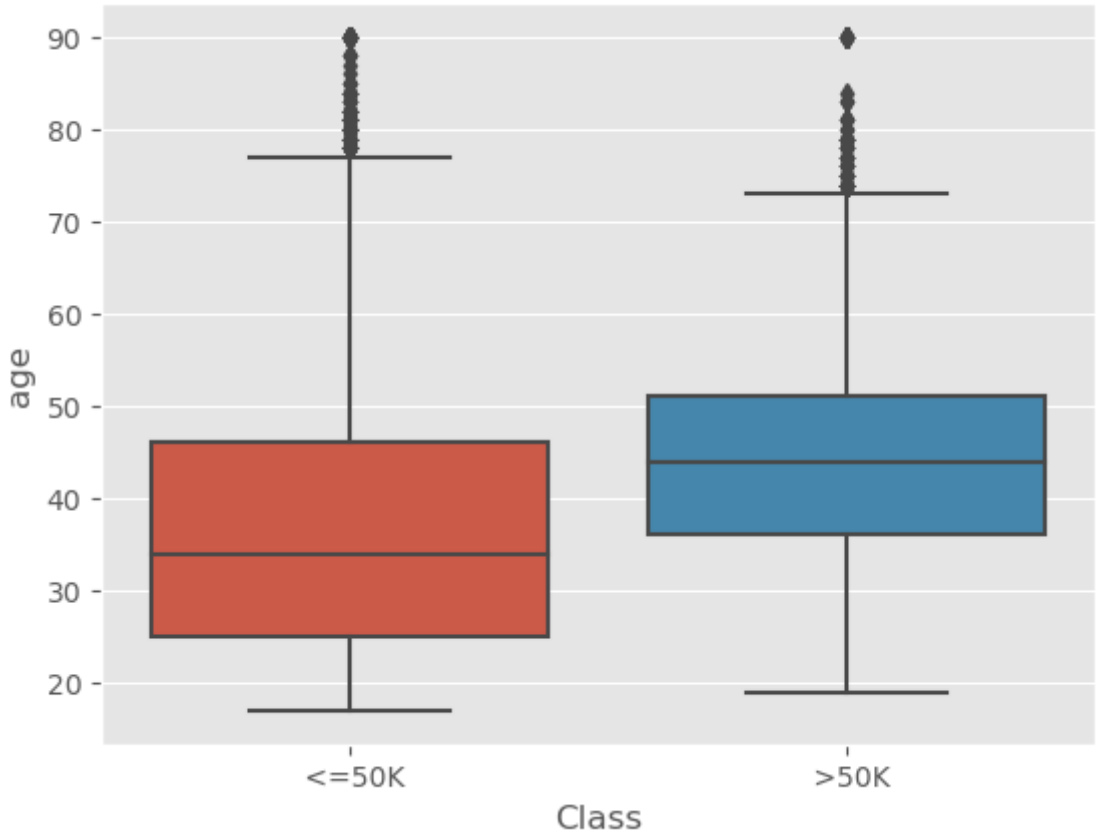
In [ ]:
```
sns.boxplot(x='Class', y='capital-loss', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='capital-loss'>
```



In [ ]:
```
sns.boxplot(x='Class', y='age', data = df)
```

Out[ ]: <AxesSubplot:xlabel='Class', ylabel='age'>



In [ ]:
```
sns.boxplot(x='Class', y='hours-per-week', data = df)
```

Out[ ]: <AxesSubplot:xlabel='Class', ylabel='hours-per-week'>