Task1DataMining

March 5, 2022

1 Adult Dataset

1.1 Overview

- Data Source: https://archive.ics.uci.edu/ml/datasets/adult
- Number of Instances: 48842
- Number of Input Features: 14
- Missing values: existent
- Prediction task: determine whether a person makes over 50K a year.

1.2 Features observations and types

1.2.1 Observations:

`workclass`: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

`fnlwgt`: final weight (number of people it is known that the entry represents)

`education`: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool

`education-num`: number indicating numerically the field given by `education` (cor(`education-num`, `education`) should be 1)

`marital-status`: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse

`occupation`: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

`relationship`: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried

`race`: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black

`sex`: Female, Male

`native-country`: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands

`salary`: >50K, <=50K

1.2.2 Types:

- Numerical: `age`, `fnlwgt`, `education-num`, `capital-gain`, `capital-loss`, `hours-per-week`
- Categorical: `workclass`, `education`, `marital-status`, `occupation`, `relationship`, `race`, `sex`, `native-country`, `salary`

2 Reading data & dealing with missing values

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.linear_model import LinearRegression
from itertools import combinations
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
import plotly.express as px
from sklearn.manifold import TSNE
from sklearn.impute import SimpleImputer

plt.rcParams.update({'figure.max_open_warning': 0})
```

[ ]:
```
col_names = ['age', 'workclass', 'fnlwgt', 'education', 'education_num',
             'marital_status', 'occupation', 'relationship', 'race', 'sex',
             'capital_gain', 'capital_loss',  'hours_per_week',␣
 ↪'native_country',
             'salary' ]
data_train = pd.read_csv('./drive/MyDrive/DataMining/T1/adult.data.csv',
                         names=col_names, header=None)
data_test = pd.read_csv('./drive/MyDrive/DataMining/T1/adult.test.csv',
                        names=col_names, header=None)

data = pd.concat([data_train, data_test], axis=0, ignore_index=True)
display(data.sample(10))
```

```
       age         workclass  fnlwgt     education  education_num  \
14447   40      Self-emp-inc   33126       Masters             14
2533    27           Private  466224  Some-college             10
46379   30           Private  131415     Bachelors             13
35175   33  Self-emp-not-inc  114639          11th              7
27971   67                 ?  102693       HS-grad              9
```

```
46371   43        State-gov  598995     Bachelors             13
12970   48     Self-emp-inc  213140  Some-college             10
6234    22     Self-emp-inc  269583       7th-8th              4
7304    23          Private  398904     Bachelors             13
20608   72                ?  166253       HS-grad              9

             marital_status         occupation    relationship    race      sex  \
14447    Married-civ-spouse       Craft-repair         Husband   White     Male
2533          Never-married              Sales   Not-in-family   Black     Male
46379         Never-married    Priv-house-serv   Not-in-family   White   Female
35175         Never-married    Farming-fishing       Unmarried   White     Male
27971               Widowed                  ?   Not-in-family   White     Male
46371    Married-civ-spouse     Prof-specialty            Wife   Black   Female
12970    Married-civ-spouse    Exec-managerial         Husband   White     Male
6234     Married-civ-spouse       Craft-repair         Husband   White     Male
7304     Married-civ-spouse     Prof-specialty            Wife   White   Female
20608    Married-civ-spouse                  ?            Wife   White   Female

       capital_gain  capital_loss  hours_per_week  native_country   salary
14447             0             0              50   United-States    <=50K
2533              0             0              40   United-States    <=50K
46379             0             0              60   United-States   <=50K.
35175             0             0              40   United-States   <=50K.
27971          1086             0              35   United-States    <=50K
46371          3103             0              40   United-States    >50K.
12970             0             0              40   United-States     >50K
6234           2580             0              40   United-States    <=50K
7304              0             0              40   United-States    <=50K
20608             0             0               2   United-States    <=50K
```

Eliminate dots from the end in salary column:

[ ]:
```python
data['salary'] = data['salary'].apply(lambda x: x.replace(".", ""))
```

[ ]:
```python
data = data.replace(to_replace ='^\s*\?\s*$', value = np.NaN, regex=True)
#print(' ?' in data.values )
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       46043 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education_num   48842 non-null  int64
 5   marital_status  48842 non-null  object
 6   occupation      46033 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital_gain    48842 non-null  int64
 11  capital_loss    48842 non-null  int64
 12  hours_per_week  48842 non-null  int64
 13  native_country  47985 non-null  object
 14  salary          48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

The proportion of null variables per columns is small, thus we opt to not eliminate rows/columns, but to replace them with mode and median values:

[ ]:
```
print(data.isna().sum()/data.shape[0] * 100)
```

```
age               0.000000
workclass         5.730724
fnlwgt            0.000000
education         0.000000
education_num     0.000000
marital_status    0.000000
occupation        5.751198
relationship      0.000000
race              0.000000
sex               0.000000
capital_gain      0.000000
capital_loss      0.000000
hours_per_week    0.000000
native_country    1.754637
salary            0.000000
dtype: float64
```

Replace NAN values of categorical features with the mode and NAN values of numerical features with median (mean is more suited for data with normal distribution):

[ ]:
```
# data['workclass'].fillna(data['workclass'].mode()[0], inplace=True)
# data['occupation'].fillna(data['occupation'].mode()[0], inplace=True)
# data['native_country'].fillna(data['native_country'].mode()[0], inplace=True)

categorical_columns = list(data.select_dtypes(include=['object']).columns)
numerical_columns = list(data.select_dtypes(include=['float64', 'int64']).
 ↪columns)

print("Categorical columns: ", categorical_columns)
print("Numerical columns: ", numerical_columns)
print()
print(data.isna().sum())
print()

# for col in categorical_columns:
#   data[col].fillna(data[col].mode()[0], inplace=True)

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(data[categorical_columns])
data[categorical_columns] = imputer.transform(data[categorical_columns])

# for col in numerical_columns:
#   data[col].fillna(data[col].median(), inplace=True)

imputer = SimpleImputer(strategy='median', missing_values=np.nan)
imputer = imputer.fit(data[numerical_columns])
data[numerical_columns] = imputer.transform(data[numerical_columns])

print(data.isna().sum())
```

```
Categorical columns:  ['workclass', 'education', 'marital_status', 'occupation',
'relationship', 'race', 'sex', 'native_country', 'salary']
Numerical columns:  ['age', 'fnlwgt', 'education_num', 'capital_gain',
'capital_loss', 'hours_per_week']

age                  0
workclass         2799
fnlwgt               0
education            0
education_num        0
marital_status       0
occupation        2809
relationship         0
race                 0
sex                  0
capital_gain         0
capital_loss         0
hours_per_week       0
native_country     857
salary               0
dtype: int64

age               0
workclass         0
fnlwgt            0
education         0
education_num     0
```

```
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
salary            0
dtype: int64
```

3 Encode categorical features based on frequency

We encode the categorical features based on their frequency (most frequent category gets values 0, second frequent category gets values 1, etc.)

We will use this encoding in analyzing the variables with predictive power for the target variables and for applying dimensionality reduction (t-SNE).

[ ]:
```
data_cat_to_num = data.copy()
for column in data_cat_to_num.columns:
  if column in categorical_columns:
    dict_col = dict(data_cat_to_num[column].value_counts() )
    value = 0
    for (feature, val) in dict_col.items():
      dict_col[feature] = value
      value+=1

    #print(dict_col)
    data_cat_to_num[column]=data_cat_to_num[column].replace(dict_col)

display(data_cat_to_num)
```

```
        age  workclass    fnlwgt  education  education_num  marital_status  \
0      39.0          3   77516.0          2           13.0               1
1      50.0          1   83311.0          2           13.0               0
2      38.0          0  215646.0          0            9.0               2
3      53.0          0  234721.0          5            7.0               0
4      28.0          0  338409.0          2           13.0               0
...     ...        ...       ...        ...            ...             ...
48837  39.0          0  215419.0          2           13.0               2
48838  64.0          0  321403.0          0            9.0               4
48839  38.0          0  374983.0          2           13.0               0
48840  44.0          0   83891.0          2           13.0               2
48841  35.0          4  182148.0          2           13.0               0

       occupation  relationship  race  sex  capital_gain  capital_loss  \
0               3             1     0    0        2174.0           0.0
```

```
1              2            0     0    0          0.0           0.0
2              8            1     0    0          0.0           0.0
3              8            0     1    0          0.0           0.0
4              0            4     1    1          0.0           0.0
...          ...          ...   ...  ...          ...           ...
48837          0            1     0    1          0.0           0.0
48838          0            5     1    0          0.0           0.0
48839          0            0     0    0          0.0           0.0
48840          3            2     2    0       5455.0           0.0
48841          2            0     0    0          0.0           0.0

       hours_per_week  native_country  salary
0                40.0               0       0
1                13.0               0       0
2                40.0               0       0
3                40.0               0       0
4                40.0               8       0
...               ...             ...     ...
48837            36.0               0       0
48838            40.0               0       0
48839            50.0               0       0
48840            40.0               0       0
48841            60.0               0       1

[48842 rows x 15 columns]
```

4 Univariate analysis

4.1 Central tendency statistics

4.1.1 Mean

Obs: * The mean of capital gain is significantly higher than the mean of capital loss.

[ ]:
```
data_numerical = data.select_dtypes(include=np.number)
print("Mean: ")
data_numerical.mean()
```

```
Mean:
```

[ ]:
```
age                   38.643585
fnlwgt            189664.134597
education_num         10.078089
capital_gain        1079.067626
capital_loss          87.502314
hours_per_week        40.422382
dtype: float64
```

4.1.2 Median

It can be observed that for features age, education_num, hours_per_week and fnlwgt the value of the median is close to the value of the mean.

[ ]:
```
print("Median: ")
data_numerical.median()
```

```
Median:
```

[ ]:
```
age                   37.0
fnlwgt            178144.5
education_num         10.0
capital_gain           0.0
capital_loss           0.0
hours_per_week        40.0
dtype: float64
```

4.1.3 Mode

The mode indicates the most frequent value(s) for each feature in the dataset.

[ ]:
```
print("Mode: ")
data.mode()
```

```
Mode:
```

[ ]:
```
   age workclass  fnlwgt  ... hours_per_week  native_country  salary
0   36   Private  203488  ...             40   United-States   <=50K

[1 rows x 15 columns]
```

4.2 Dispersion statistics

- Absolute amplitude: $R = maxvalue - minvalue$
- Relative amplitude: $R\% = \frac{R}{mean} * 100$
- Interquartile range: $IQR = Q3 - Q1$
- Variance: variance = $variance = \sum_{i=1}^{n}(x_i - mean)^2/(n-1)$
- Standard deviation: $std = \sqrt{variance}$

[ ]:
```
data_stat = data.describe().T
data_stat["abs_amplitude"] = data_stat["max"] - data_stat["min"]
data_stat["rel_amplitude"] = data_stat["abs_amplitude"] / data_stat["mean"] *␣
 ↪100
data_stat["interquartile_range"] = data_stat["75%"] - data_stat["25%"]
data_stat["variance"] = data_stat["std"] * data_stat["std"]

data_stat = data_stat.drop(columns= ['count', 'mean', 'min', 'max', '50%'])
```

```
display(data_stat.rename(columns={'25%': 'Q1', '75%': 'Q3'}))
```

```
                        std        Q1  ...  interquartile_range      variance
age               13.710510      28.0  ...                 20.0  1.879781e+02
fnlwgt        105604.025423  117550.5  ...             120091.5  1.115221e+10
education_num      2.570973       9.0  ...                  3.0  6.609901e+00
capital_gain    7452.019058       0.0  ...                  0.0  5.553259e+07
capital_loss     403.004552       0.0  ...                  0.0  1.624127e+05
hours_per_week    12.391444      40.0  ...                  5.0  1.535479e+02

[6 rows x 7 columns]
```

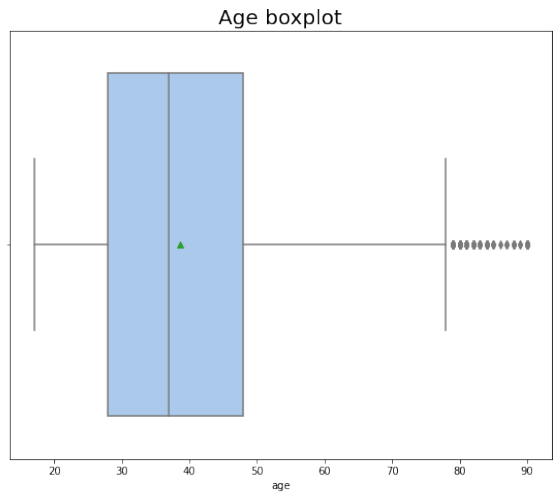
4.3 Boxplots

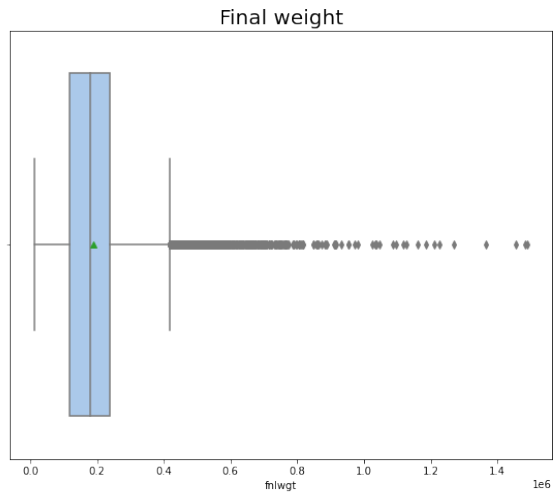
4.3.1 Simple boxplots

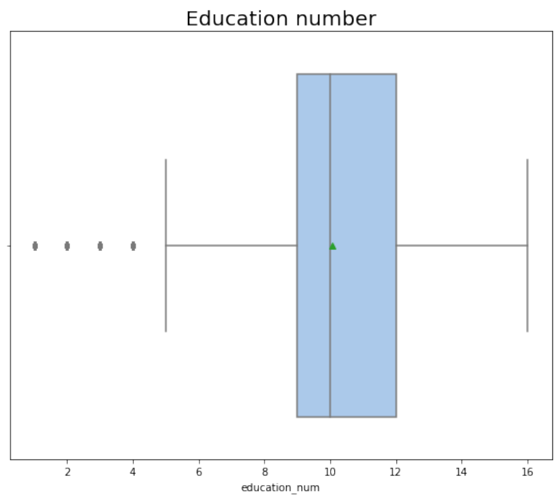
In order to interpret boxplots: - the box represents the middle 50% values of the feature - the black central vertical line indicates the median - Q1 and Q3 are the vertical lines that represent the margins of the blue rectangle - the width of the blue rectangle is IQR (Q3-Q1)
- the small green triangle indicates the mean - the leftmost and rightmost vertical lines indicate the "minimum non-outlier" (Q1-1.5*IQR) and the "maximum non-outlier" (Q3+1.5*IQR) - the black dots represent the outliers

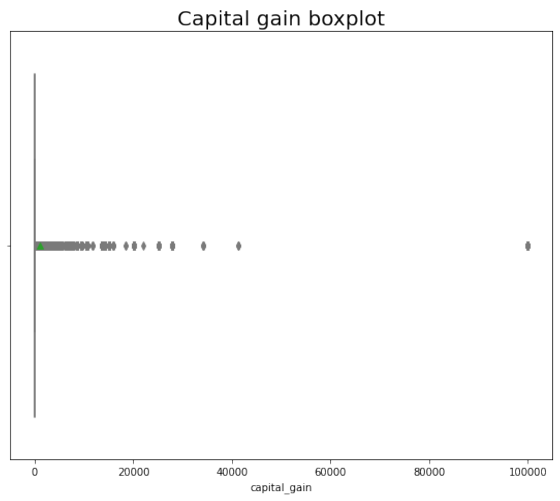
Obs: - Distributions of features `age` and `fnlwgt` are skewed right (we can deduce this by the box, whiskers and outliers positions) - Distributions of features `capital_gain` and `capital_loss` are clearly leptokurtic (due to the fact that the box is inexistent) - Distribution with many outliers are `fnlwgt`, `capital_gain`, `capital_loss` and `hours_per_week`

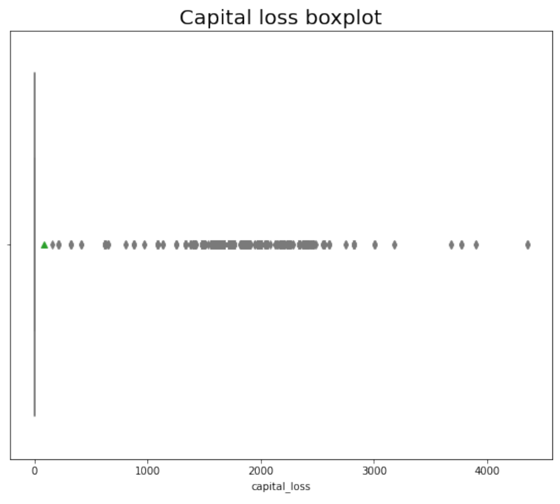
[ ]:
```
fig, axs = plt.subplots(3, 2, figsize=(15,20), constrained_layout = True)
for idx in range(0,6):
    p=sns.boxplot(data=data, x=numerical_columns[idx], ax=axs[idx//2, idx%2],
                  palette="pastel", showmeans=True)
    title = str(numerical_columns[idx]).capitalize().replace("_", " ")
    title += " boxplot"
    if numerical_columns[idx] == "education_num":
        title = "Education number"
    elif numerical_columns[idx] == "fnlwgt":
        title = "Final weight"

    p.set_title(title, fontsize=20)
```

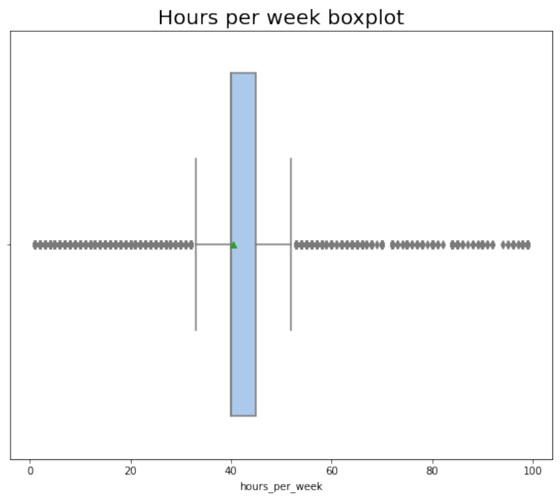

Age boxplot
20
30
40
50
60
70
80
90
age


Final weight
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
fnlwgt
1e6


Education number
2
4
6
8
10
12
14
16
education_num


Capital gain boxplot
0
20000
40000
60000
80000
100000
capital_gain


Capital loss boxplot
0
1000
2000
3000
4000
capital_loss

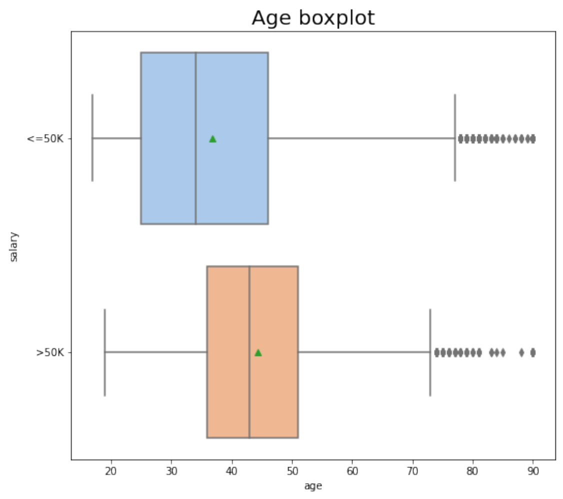

Hours per week boxplot
0
20
40
60
80
100
hours_per_week

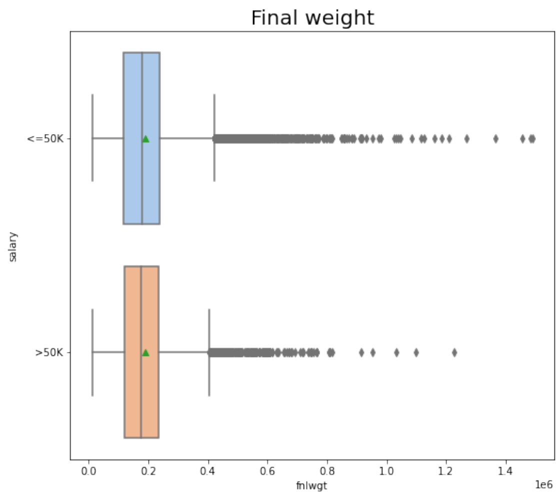
4.3.2 Boxplots by salary

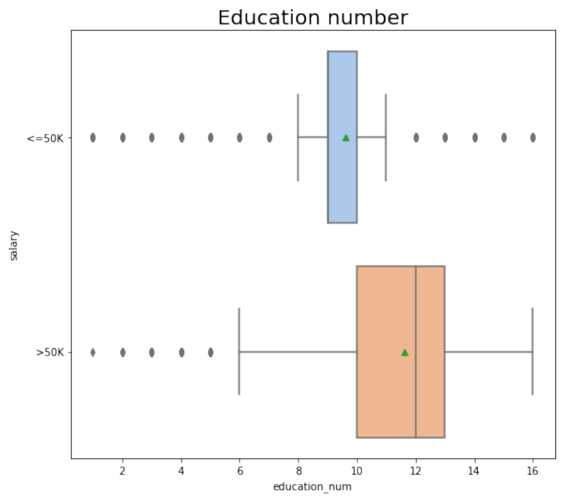
The following boxplots are distributed by the values of salary (<=50K, >50K). Interpreting these boxplots, we can observe that: - (almost) 50% of people with a salary >50K have an age between 35 and 50 (whereas 50% of people with a salary <=50K have an age between 25 and 45)

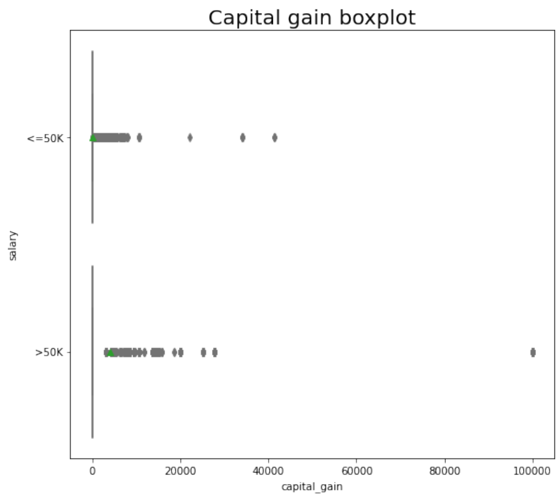
- (almost) 50% of people with a salary >50K have finished some college/ vocational college/ obtained their Bachelors degree/finished an Associate Academy, but do not have a Masters or Doctorate degree. (whereas 50% of people with a salary <=50K have finished high school or some college)
- (almost) 50% of people with a salary >50K work between 40 and 50 hours a week (whereas 50% of people with a salary <=50K work at most 40 hours a week)

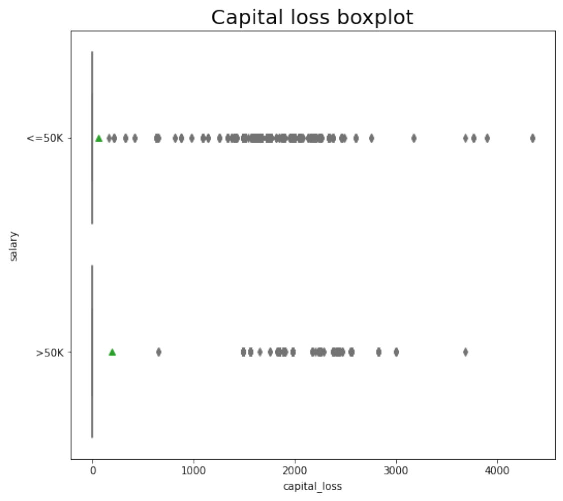
[ ]:
```
fig, axs = plt.subplots(3, 2, figsize=(15,20), constrained_layout = True)
for idx in range(0,6):
    p=sns.boxplot(data=data, x=numerical_columns[idx], ax=axs[idx//2, idx%2],
                  y="salary", palette="pastel", showmeans=True)
    title = str(numerical_columns[idx]).capitalize().replace("_", " ")
    title += " boxplot"
    if numerical_columns[idx] == "education_num":
        title = "Education number"
    elif numerical_columns[idx] == "fnlwgt":
        title = "Final weight"

    p.set_title(title, fontsize=20)
```

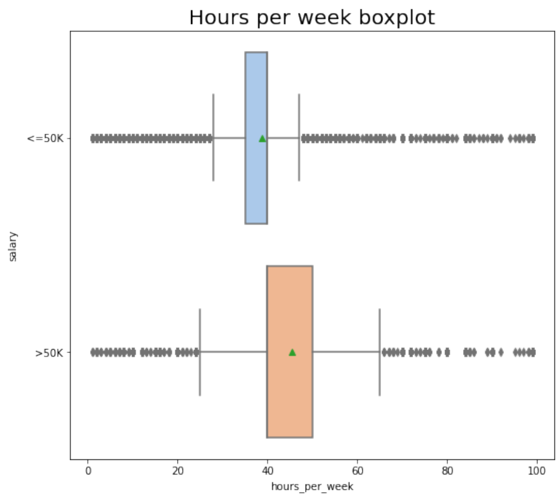

Age boxplot
<=50K
>50K
salary
20 30 40 50 60 70 80 90
age


Final weight
<=50K
>50K
salary
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
1e6
fnlwgt


Education number
<=50K
>50K
salary
2 4 6 8 10 12 14 16
education_num


Capital gain boxplot
<=50K
>50K
salary
0 20000 40000 60000 80000 100000
capital_gain


Capital loss boxplot
<=50K
>50K
salary
0 1000 2000 3000 4000
capital_loss


Hours per week boxplot
<=50K
>50K
salary
0 20 40 60 80 100
hours_per_week

4.4 Form of distribution and density plots

4.4.1 Skewness

To interpret skewness:

1.

- skewness between -0.5, 0.5 => data almost symmetrical
- skewness between -1, -0.5 or 0.5, 1 => data moderately skewed
- skewness < -1 or > 1 => high skewness

1.

- positive skweness => right skewed distribution
- negative skweness => left skewed distribution

[ ]:
```
skewness = data_numerical.skew(axis=0)
print(skewness)
```

```
age                0.557580
fnlwgt             1.438892
education_num     -0.316525
capital_gain      11.894659
capital_loss       4.569809
hours_per_week     0.238750
dtype: float64
```

4.4.2 Kurtosis

To interpret kurtosis:

- kurtosis > 3 => leptokurtic distribution
- kurtosis $\approx$ 3 => mesokurtic distribution
- kurtosis < 3 => platykurtic

[ ]:
```
kurtosis = data_numerical.kurt(axis=0)
print(kurtosis)
```

```
age                -0.184269
fnlwgt              6.057848
education_num       0.625745
capital_gain      152.693096
capital_loss       20.014346
hours_per_week      2.951059
dtype: float64
```

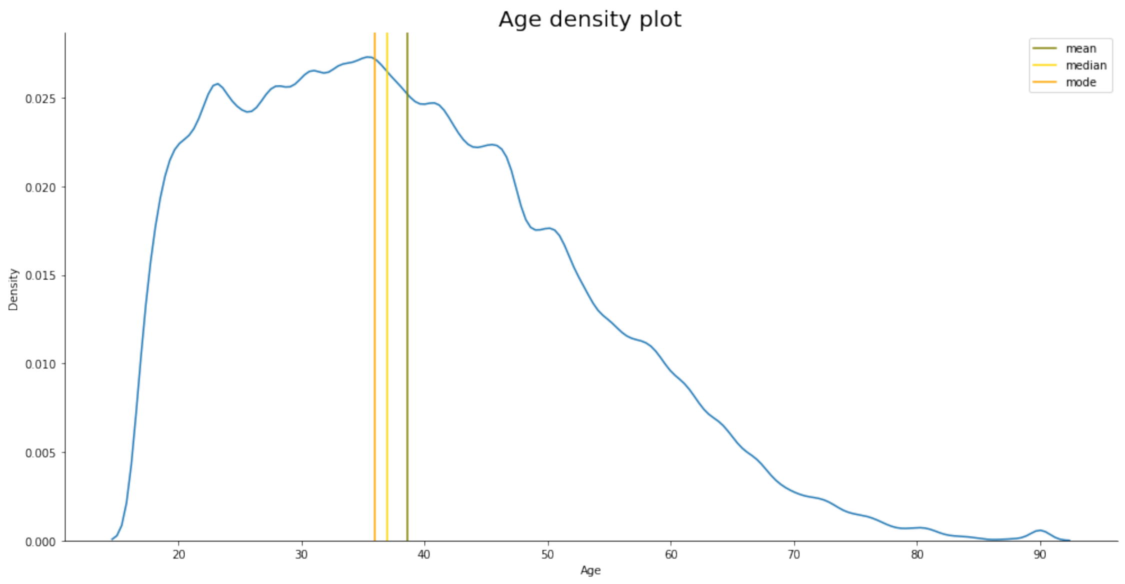
4.4.3 Density plots

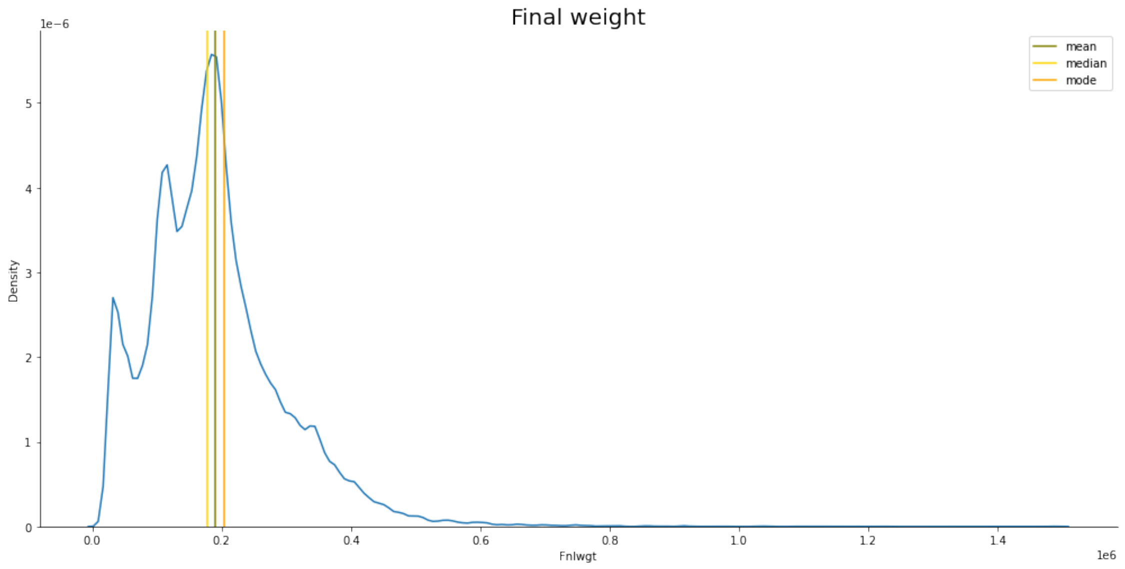
Interpretation:

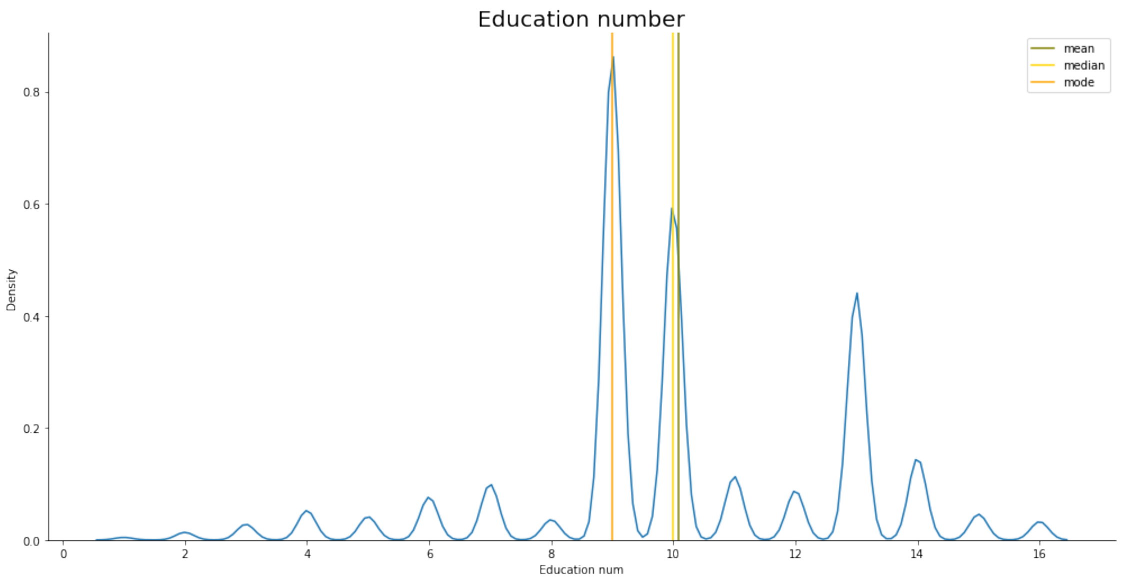
- Skewness:
- Distributions for `fnlwgt`, `capital_gain`, `capital_loss` are highly right skewed.
- Distributuion for age is moderately right skewed
- Distributions for `hours_per_week` and `education_num` are almost symmetrical
- Kurtosis:
- Distributions for `fnlwgt`, `capital_gain` and `capital_loss` are leptokurtic.
- Distribution for `hours_per_week` is mesokurtic
- Distributions for age and `education_num` are platykurtic

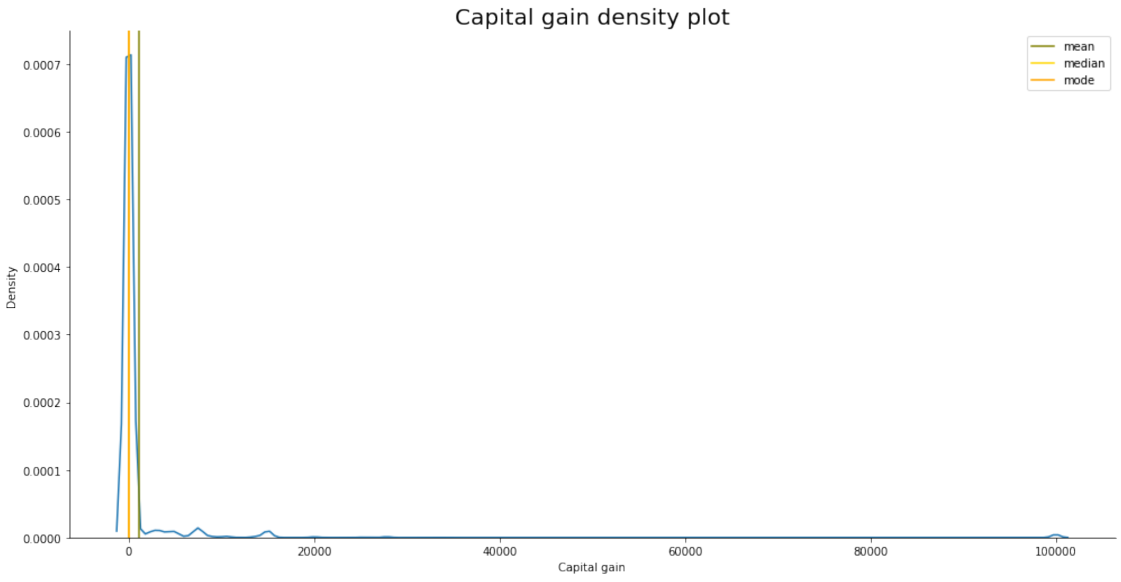
[ ]:
```
data_numerical = data.select_dtypes(include=np.number)
for idx in range(0,6):

    p = sns.displot(data_numerical, x=numerical_columns[idx], kind="kde",␣
 ↪height=7, aspect=2, bw_adjust=0.5)

    title = str(numerical_columns[idx]).capitalize().replace("_", " ")
    title += " density plot"
    if numerical_columns[idx] == "education_num":
        title = "Education number"
    elif numerical_columns[idx] == "fnlwgt":
        title = "Final weight"

    plt.title(title, fontsize=20)
    p.set_axis_labels(str(numerical_columns[idx]).capitalize().replace("_", " "),
                      "Density")

    plt.axvline(x=data_numerical.mean()[numerical_columns[idx]], color="olive",␣
 ↪label="mean")
    plt.axvline(x=data_numerical.median()[numerical_columns[idx]], color="gold"␣
 ↪, label="median")
    plt.axvline(x=data_numerical.mode()[numerical_columns[idx]].values[0],␣
 ↪color="orange", label="mode")
    plt.legend()

# p = sns.displot(data_numerical, x="fnlwgt", kind="kde", height=10)
# plt.axvline(x=data_numerical.mean()["fnlwgt"], color="olive", label="mean")
# plt.axvline(x=data_numerical.median()["fnlwgt"], color="gold" ,␣
 ↪label="median")
# plt.axvline(x=data_numerical.mode()["fnlwgt"].values[0], color="orange",␣
 ↪label="mode")
```

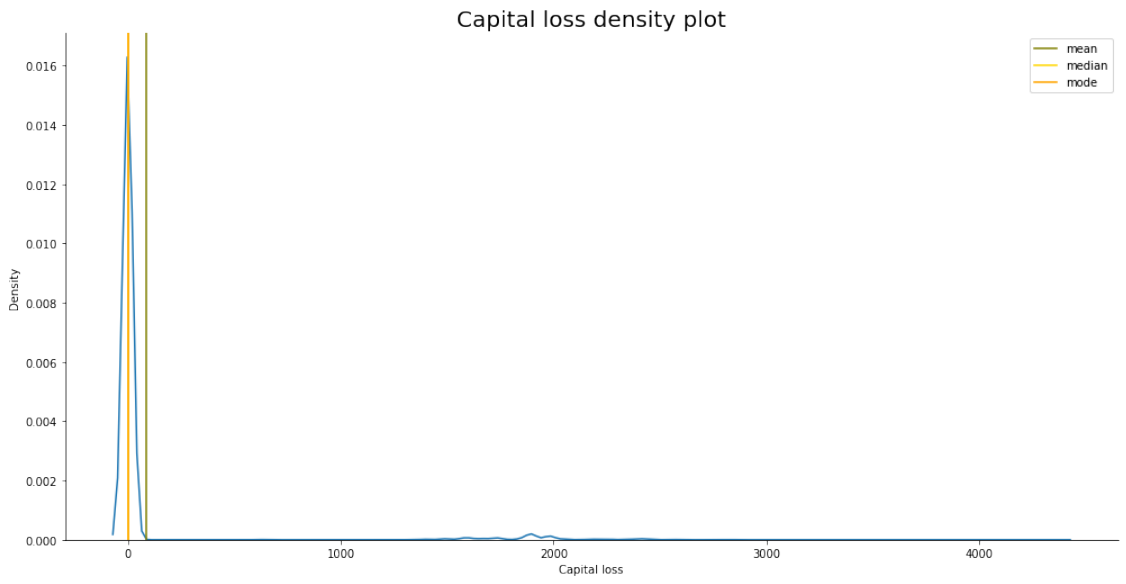
```
# plt.legend()
```

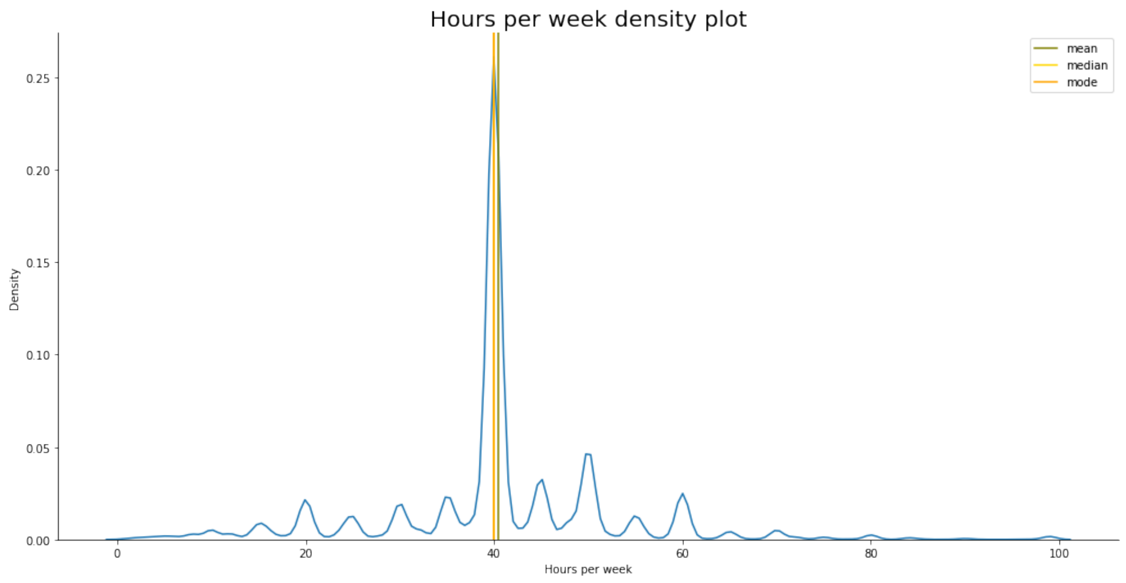

Age density plot
mean
median
mode
Density
0.000
0.005
0.010
0.015
0.020
0.025
20
30
40
50
60
70
80
90
Age


Final weight
1e−6
mean
median
mode
Density
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
Fnlwgt


Education number
mean
median
mode
Density
Education num
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10
12
14
16


Capital gain density plot
mean
median
mode
Density
Capital gain
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0
20000
40000
60000
80000
100000





4.5 Frequencies for categorical attributes and histograms

Number of values of categories (unique) and mode (top) with associated frequency for each category:

[ ]:
```
data_categorical = data.select_dtypes(include=['object'])
display(data_categorical.describe().T)
```

```
               count unique                 top   freq
workclass      48842      8             Private  36705
education      48842     16             HS-grad  15784
marital_status 48842      7  Married-civ-spouse  22379
occupation     48842     14      Prof-specialty   8981
relationship   48842      6             Husband  19716
race           48842      5               White  41762
sex            48842      2                Male  32650
native_country 48842     41       United-States  44689
salary         48842      2                <=50K  37155
```

4.5.1 Frequencies for categorical attributes

[ ]:
```
for col in categorical_columns:
  count = data.groupby(col).size()
  print("Frequencies for ", col, " attribute: ")
  display(count)
  print()
```

```
Frequencies for  workclass  attribute:

workclass
 Federal-gov          1432
 Local-gov            3136
 Never-worked           10
 Private             36705
 Self-emp-inc         1695
 Self-emp-not-inc     3862
 State-gov            1981
 Without-pay            21
dtype: int64



Frequencies for  education  attribute:

education
 10th             1389
 11th             1812
 12th              657
 1st-4th           247
 5th-6th           509
 7th-8th           955
 9th               756
 Assoc-acdm       1601
 Assoc-voc        2061
 Bachelors        8025
 Doctorate         594
 HS-grad         15784
 Masters          2657
 Preschool          83
 Prof-school       834
 Some-college    10878
dtype: int64



Frequencies for  marital_status  attribute:

marital_status
 Divorced                  6633
 Married-AF-spouse           37
 Married-civ-spouse       22379
 Married-spouse-absent      628
 Never-married            16117
 Separated                 1530
 Widowed                   1518
dtype: int64



Frequencies for  occupation  attribute:

occupation
 Adm-clerical         5611
 Armed-Forces           15
 Craft-repair         6112
 Exec-managerial      6086
 Farming-fishing      1490
 Handlers-cleaners    2072
 Machine-op-inspct    3022
 Other-service        4923
 Priv-house-serv       242
 Prof-specialty       8981
 Protective-serv       983
 Sales                5504
 Tech-support         1446
 Transport-moving     2355
dtype: int64



Frequencies for  relationship  attribute:

relationship
 Husband           19716
 Not-in-family     12583
 Other-relative     1506
 Own-child          7581
 Unmarried          5125
 Wife               2331
dtype: int64



Frequencies for  race  attribute:

race
 Amer-Indian-Eskimo      470
 Asian-Pac-Islander     1519
 Black                  4685
 Other                   406
 White                 41762
dtype: int64



Frequencies for  sex  attribute:

sex
 Female    16192
 Male      32650
dtype: int64



Frequencies for  native_country  attribute:

native_country
 Cambodia                        28
 Canada                         182
 China                          122
 Columbia                        85
 Cuba                           138
 Dominican-Republic             103
 Ecuador                         45
 El-Salvador                    155
 England                        127
 France                          38
 Germany                        206
 Greece                          49
 Guatemala                       88
 Haiti                           75
 Holand-Netherlands               1
 Honduras                        20
 Hong                           30
 Hungary                        19
 India                         151
 Iran                           59
 Ireland                        37
 Italy                         105
 Jamaica                       106
 Japan                          92
 Laos                           23
 Mexico                        951
 Nicaragua                      49
 Outlying-US(Guam-USVI-etc)     23
 Peru                           46
 Philippines                   295
 Poland                         87
 Portugal                       67
 Puerto-Rico                   184
 Scotland                       21
 South                         115
 Taiwan                         65
 Thailand                       30
 Trinadad&Tobago                27
 United-States               44689
 Vietnam                        86
 Yugoslavia                     23
dtype: int64



Frequencies for  salary  attribute:

salary
 <=50K    37155
 >50K     11687
dtype: int64
```

4.5.2 Histograms for categorical & numerical features, taking into account the frequency by salary.

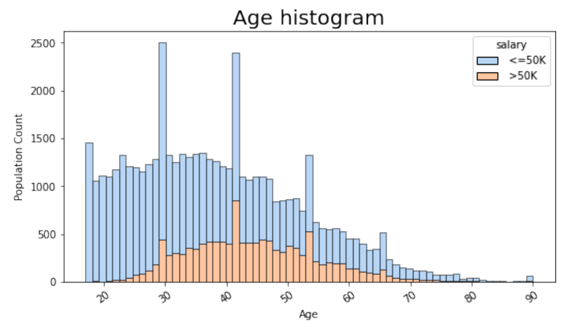
Obs: Number of chosen bins for numerical feature education_num equals number of education levels (16).

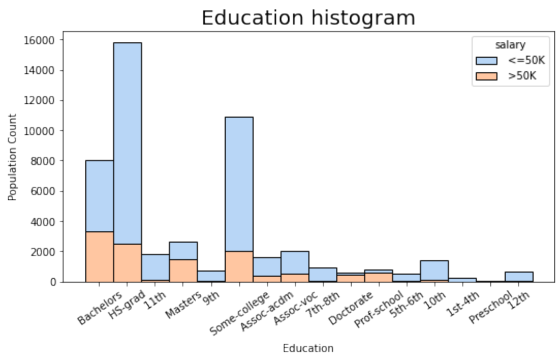
Interpretations for categorical features: - regarding `workclass`, most people work in the Private sector - most of the population have either attended some college or they are high school graduates (`education`) - regarding `marital_status`, most people are either married to a civil spouse, or they have never been married - regarding `race`, the most frequent one is the White race - the number of men is almost two times the number of women (`sex` feature); this may explain why in case of `relationship` feature the value 'husband' is by far the most frequent one - the most frequent native country is United States of America (`native_country` feature)

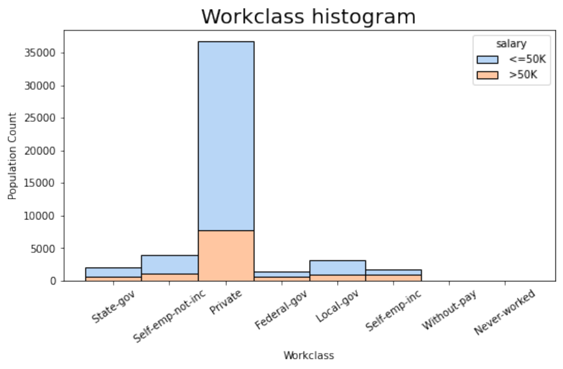
Interpretations for numerical features:

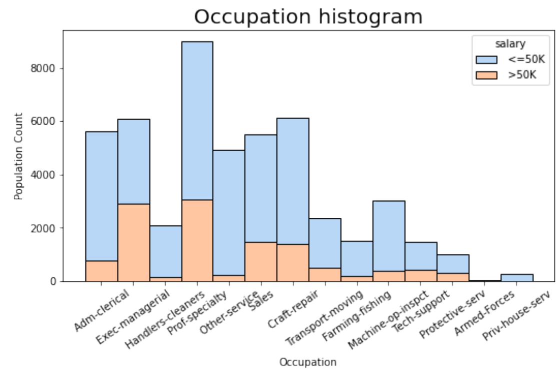
- most common values for `hours_per_week` range from 36 to 40 (inclusively)
- as it can be expected there is a direct correlation between `education num` and `education_features`
- in case of `capital_gain` and `capital_loss` features most sum values that are gained or lost are relatively small

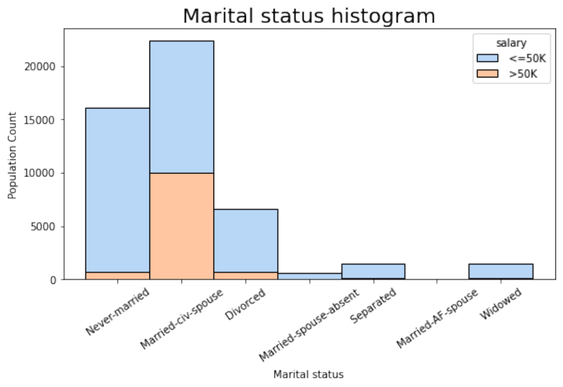
[ ]:
```
fig, axs = plt.subplots(4, 2, figsize=(15,20), constrained_layout = True)

features = ["age", "education", "workclass", "occupation",
            "marital_status", "relationship", "race", "sex"]

for idx in range(0,8):
  p = sns.histplot(data=data, ax=axs[idx//2, idx%2], stat="count",
              multiple="stack", palette="pastel", element="bars", legend=True,
              x=features[idx], hue="salary"
              )
  p.set_title(str(features[idx]).capitalize().replace("_", " ")+" histogram",
              fontsize=20)
  p.set_xlabel(str(features[idx]).capitalize().replace("_", " "))
  p.set_ylabel("Population Count")
  p.tick_params(axis='x', rotation=35)
plt.show()

f = plt.figure(figsize=(25,5))
ax = f.add_subplot(1,1,1)
p_native_country =  sns.histplot(data=data, ax=ax, stat="count",␣
 ↪multiple="stack",
                    palette="pastel",element="bars", legend=True,
                    x="native_country", hue="salary"
                    )
p_native_country.set_title("Native country histogram")
p_native_country.set_xlabel("Native Country")
p_native_country.set_ylabel("Population Count")
p_native_country.tick_params(axis='x', rotation=35)

plt.show()

num_features = ["fnlwgt", "education_num", "capital_gain",
                "capital_loss", "hours_per_week" ] #except age

fig, axs = plt.subplots(3, 2, figsize=(15,20), constrained_layout = True)
for idx in range(0,5):
   nb_bins = 25
   if (num_features[idx] == "education_num"):
      nb_bins = 16
   p = sns.histplot(data=data, bins=nb_bins,
                    ax=axs[idx//2, idx%2], stat="count", multiple="stack",
                    palette="pastel", element="bars", legend=True,
                    x=num_features[idx], hue="salary")
   title = str(num_features[idx]).capitalize().replace("_", " ") + " histogram"
   if num_features[idx] == "education_num":
       title = "Education number"
   elif num_features[idx] == "fnlwgt":
       title = "Final weight"
   p.set_title(title, fontsize=20)
   p.set_xlabel(str(num_features[idx]).capitalize().replace("_", " "))
   p.set_ylabel("Population Count")

fig.delaxes(axs[2, 1])
plt.show()
```

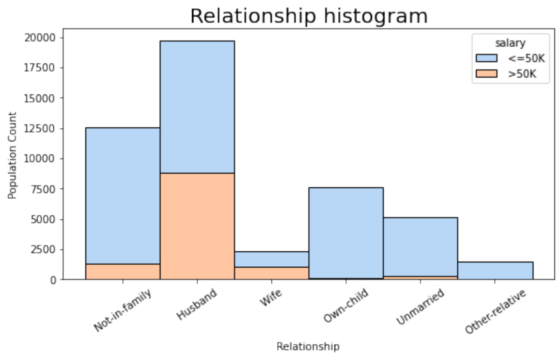

Age histogram
salary
<=50K
>50K
Population Count
Age
0
500
1000
1500
2000
2500
20
30
40
50
60
70
80
90

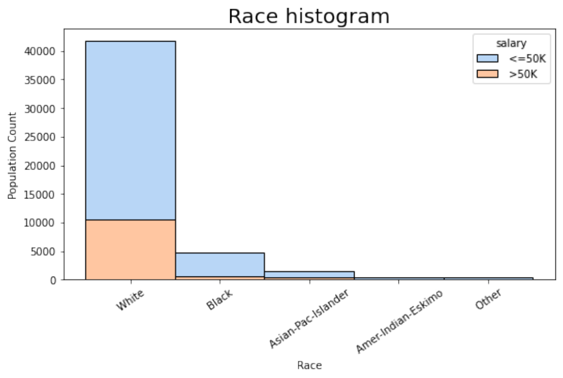

Education histogram
salary
<=50K
>50K
Population Count
Education
0
2000
4000
6000
8000
10000
12000
14000
16000
Bachelors
HS-grad
11th
Masters
9th
Some-college
Assoc-acdm
Assoc-voc
7th-8th
Doctorate
Prof-school
5th-6th
10th
1st-4th
Preschool
12th

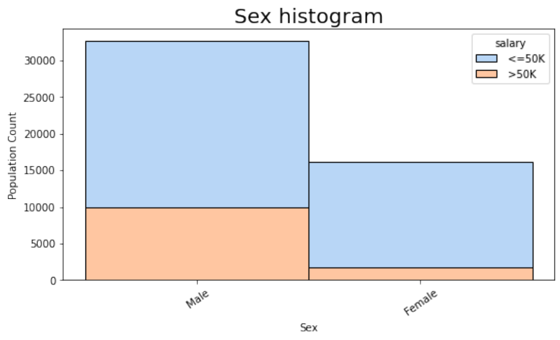

Workclass histogram
salary
<=50K
>50K
Population Count
Workclass
0
5000
10000
15000
20000
25000
30000
35000
State-gov
Self-emp-not-inc
Private
Federal-gov
Local-gov
Self-emp-inc
Without-pay
Never-worked

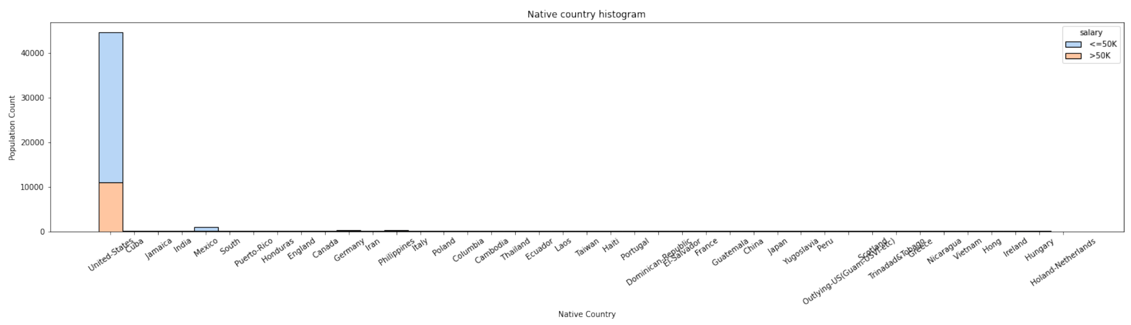

Occupation histogram
salary
<=50K
>50K
Population Count
Occupation
0
2000
4000
6000
8000
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Craft-repair
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
Protective-serv
Armed-Forces
Priv-house-serv

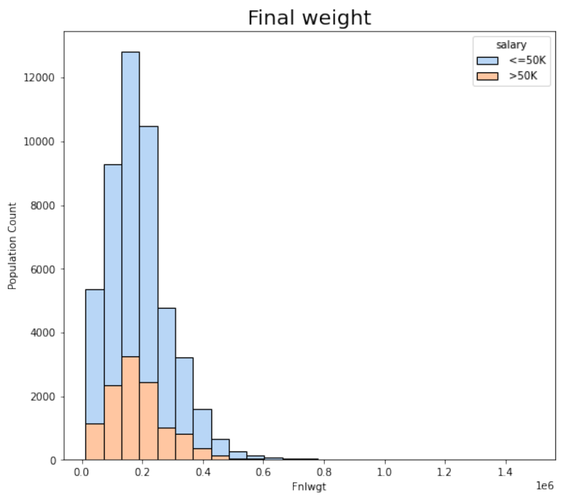
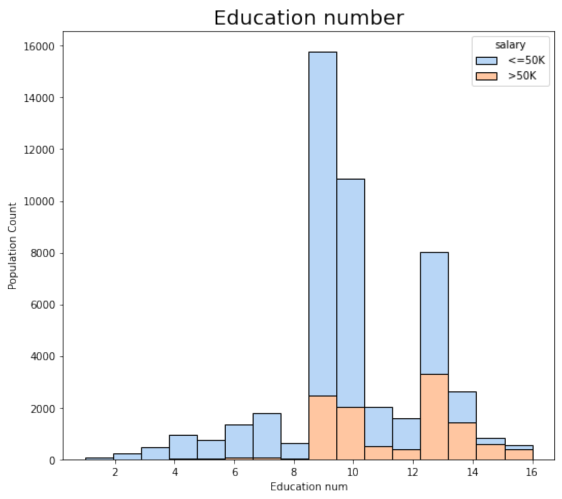
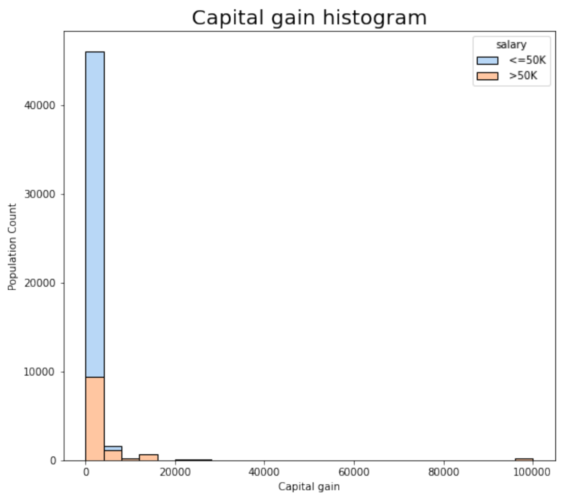
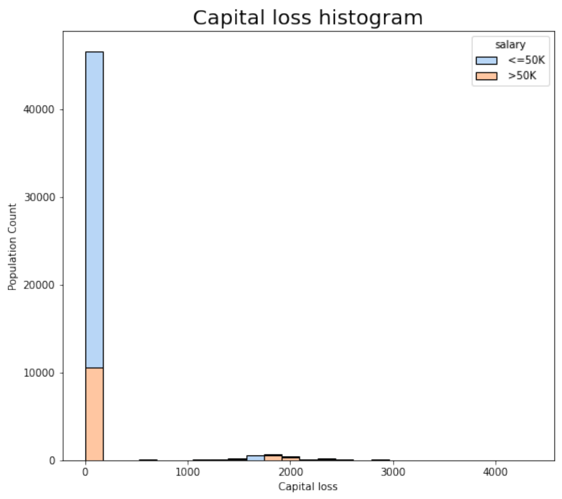
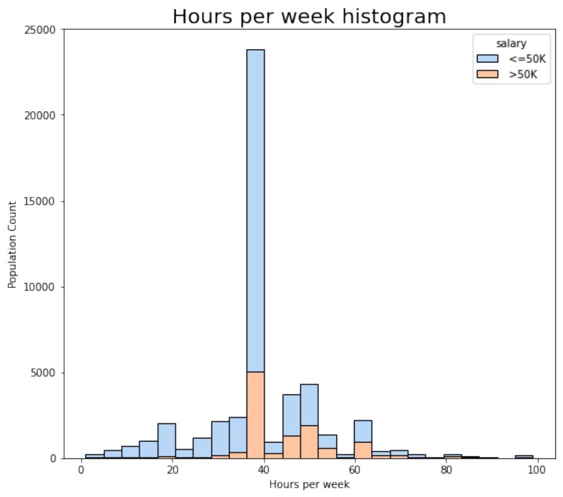

Marital status histogram
salary
<=50K
>50K
Population Count
Marital status
0
5000
10000
15000
20000
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed


Relationship histogram
salary
<=50K
>50K
Population Count
Relationship
0
2500
5000
7500
10000
12500
15000
17500
20000
Not-in-family
Husband
Wife
Own-child
Unmarried
Other-relative


Race histogram
salary
<=50K
>50K
Population Count
Race
0
5000
10000
15000
20000
25000
30000
35000
40000
White
Black
Asian-Pac-Islander
Amer-Indian-Eskimo
Other


Sex histogram
salary
<=50K
>50K
Population Count
Sex
0
5000
10000
15000
20000
25000
30000
Male
Female


Native country histogram
Population Count
0
10000
20000
30000
40000
salary
<=50K
>50K
Native Country
United-States
Cuba
Jamaica
India
Mexico
South
Puerto-Rico
Honduras
England
Canada
Germany
Iran
Philippines
Italy
Poland
Columbia
Cambodia
Thailand
Ecuador
Laos
Taiwan
Haiti
Portugal
Dominican-Republic
El-Salvador
France
Guatemala
China
Japan
Yugoslavia
Peru
Outlying-US(Guam-USVI-etc)
Scotland
Trinadad&Tobago
Greece
Nicaragua
Vietnam
Hong
Ireland
Hungary
Holand-Netherlands


Final weight
salary
<=50K
>50K
Population Count
Fnlwgt
1e6

Education number
salary
<=50K
>50K
Population Count
Education num

Capital gain histogram
salary
<=50K
>50K
Population Count
Capital gain

Capital loss histogram
salary
<=50K
>50K
Population Count
Capital loss

Hours per week histogram
salary
<=50K
>50K
Population Count
Hours per week

5 Multivariate analysis

5.0.1 Correlations and corrgrams

To interpret correlations:

- -1.0, perfect inverse correlation
- 1.0, perfect direct correlation
- ≈ 0, no correlation between variables

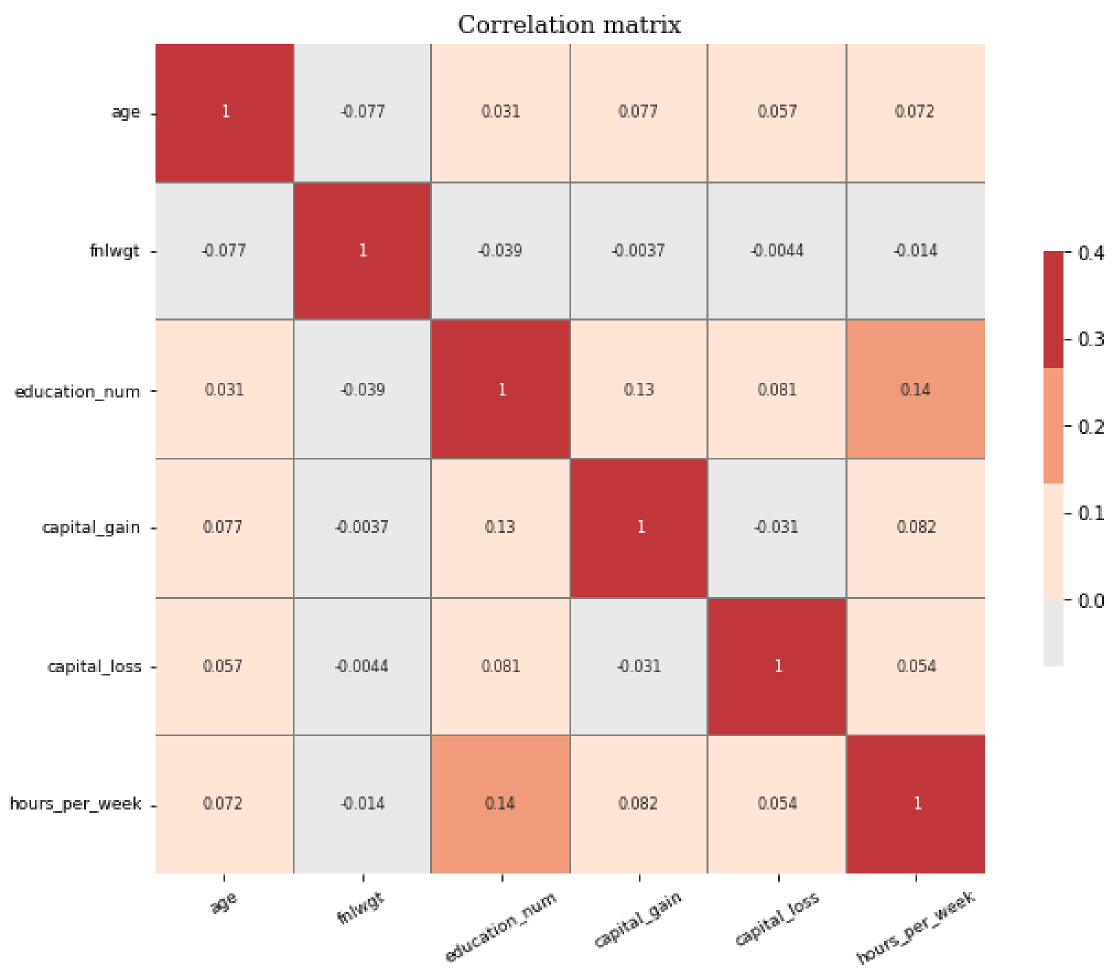
Obs: for values greater than +/- 0.5 the correlation is considered high

Table below illustrates correlations for numerical features. Correlation between all these pairs of features is small (< +/- 0.15).

[ ]:
```
correlations = data.corr()
display(correlations)
```

```
                     age    fnlwgt  ...  capital_loss  hours_per_week
age             1.000000 -0.076628  ...      0.056944        0.071558
fnlwgt         -0.076628  1.000000  ...     -0.004366       -0.013519
education_num   0.030940 -0.038761  ...      0.080972        0.143689
capital_gain    0.077229 -0.003706  ...     -0.031441        0.082157
capital_loss    0.056944 -0.004366  ...      1.000000        0.054467
hours_per_week  0.071558 -0.013519  ...      0.054467        1.000000

[6 rows x 6 columns]
```

[ ]:
```
f, ax = plt.subplots(figsize=(11, 9))
pal = sns.color_palette("RdGy_r")
cmap = pal
sns.heatmap(correlations, cmap=cmap, vmax=0.4, center=0, annot=True,
            annot_kws={"size":8}, square=True, linewidths=0.05,
            linecolor='grey', cbar=True, mask=False, cbar_kws={"shrink": 0.5})

plt.title('Correlation matrix', fontsize=13, fontfamily='serif')
plt.subplots_adjust(bottom=0.20,top=0.90,right=0.90, left=0.10)
plt.xticks(rotation=30, size=9)
plt.yticks(rotation=0, size=9)
plt.show()
```



5.0.2 Chi-squared test of independence

This test checks whether two categorical features are correlated or not. It uses the contingency table of the two features (observed counts) in order to calculate the expected frequencies.

The features are considered independent if the observed and expected frequencies are similar. (Because the expected frequencies are calculated based on the marginal probabilities assuming that the variables are independent)

The Chi2 score is a measure of the similarity between these two types of frequencies. Given the degrees of freedom (calculated having the number of rows and columns of the frequency tables) and the chi2 score we calculate de p-value. The p-value is the probability that the real chi2 value is at least as big as the calculated chi2 score.

If the p-value is smaller than a level of significance (0.05 usually) then we can reject the null hypothesis, resulting that the two features are correlated.

One requirement of applying the chi2 score test is that the expected counts in each cell are > 5 (source: https://www.theanalysisfactor.com/chi-square-test-of-independence-rule-of-thumb/).

According to chi-squared test, all pairs of categorical features for which the test can be performed are correlated.

```
[ ]: for i in range (0, len(categorical_columns)):
       for j in range(i+1, len(categorical_columns)):

           crosstab = pd.crosstab(data[categorical_columns[i]],
                                  data[categorical_columns[j]])

           #print("Observed counts: ", crosstab)
           chi2, p, dof, expected = stats.chi2_contingency(crosstab)
           #print("Expected counts: ", expected)

           if(np.count_nonzero(expected <= 5) == 0):
             probab = 0.95
             critical_val = stats.chi2.ppf(probab, dof)

             if(abs(chi2) >= critical_val):
               print("Features ",categorical_columns[i],categorical_columns[j],
                     "are correlated ( chi2 score:", chi2,", p-value:", p, ")")
             else:
               print("Features ",categorical_columns[i],categorical_columns[j],
                     "are NOT correlated ( chi2 score:", chi2, ", p-value:", p,␣
     ↪")")

             if(p <= 1 - probab):
               print("Features ",categorical_columns[i],categorical_columns[j],
                     "are correlated ( chi2 score:", chi2,", p-value:", p, ")")
             else:
               print("Features ",categorical_columns[i],categorical_columns[j],
                     "are NOT correlated ( chi2 score:", chi2, ", p-value:", p,␣
     ↪")")
             print()

           else:
               print("Test cannot be performed for features␣
     ↪",categorical_columns[i],categorical_columns[j])
```

```
Test cannot be performed for features  workclass education
Test cannot be performed for features  workclass marital_status
Test cannot be performed for features  workclass occupation
Test cannot be performed for features  workclass relationship
Test cannot be performed for features  workclass race
Test cannot be performed for features  workclass sex
Test cannot be performed for features  workclass native_country
Test cannot be performed for features  workclass salary
Test cannot be performed for features  education marital_status
Test cannot be performed for features  education occupation
Test cannot be performed for features  education relationship
Test cannot be performed for features  education race
```

```
Features  education sex are correlated ( chi2 score: 424.7053110654286 ,
p-value: 4.401401289798295e-81 )
Features  education sex are correlated ( chi2 score: 424.7053110654286 ,
p-value: 4.401401289798295e-81 )

Test cannot be performed for features  education native_country
Features  education salary are correlated ( chi2 score: 6537.972961360963 ,
p-value: 0.0 )
Features  education salary are correlated ( chi2 score: 6537.972961360963 ,
p-value: 0.0 )

Test cannot be performed for features  marital_status occupation
Test cannot be performed for features  marital_status relationship
Test cannot be performed for features  marital_status race
Features  marital_status sex are correlated ( chi2 score: 10310.411959382529 ,
p-value: 0.0 )
Features  marital_status sex are correlated ( chi2 score: 10310.411959382529 ,
p-value: 0.0 )

Test cannot be performed for features  marital_status native_country
Features  marital_status salary are correlated ( chi2 score: 9816.015037266438 ,
p-value: 0.0 )
Features  marital_status salary are correlated ( chi2 score: 9816.015037266438 ,
p-value: 0.0 )

Test cannot be performed for features  occupation relationship
Test cannot be performed for features  occupation race
Test cannot be performed for features  occupation sex
Test cannot be performed for features  occupation native_country
Test cannot be performed for features  occupation salary
Features  relationship race are correlated ( chi2 score: 1857.666018360709 ,
p-value: 0.0 )
Features  relationship race are correlated ( chi2 score: 1857.666018360709 ,
p-value: 0.0 )

Features  relationship sex are correlated ( chi2 score: 20416.778773582075 ,
p-value: 0.0 )
Features  relationship sex are correlated ( chi2 score: 20416.778773582075 ,
p-value: 0.0 )

Test cannot be performed for features  relationship native_country
Features  relationship salary are correlated ( chi2 score: 10088.722490152224 ,
p-value: 0.0 )
Features  relationship salary are correlated ( chi2 score: 10088.722490152224 ,
p-value: 0.0 )

Features  race sex are correlated ( chi2 score: 634.3972432612011 , p-value:
5.5601595323158345e-136 )
```

```
Features  race sex are correlated ( chi2 score: 634.3972432612011 , p-value:
5.5601595323158345e-136 )

Test cannot be performed for features  race native_country
Features  race salary are correlated ( chi2 score: 487.026286837627 , p-value:
4.284377710223499e-104 )
Features  race salary are correlated ( chi2 score: 487.026286837627 , p-value:
4.284377710223499e-104 )

Test cannot be performed for features  sex native_country
Features  sex salary are correlated ( chi2 score: 2248.847679013691 , p-value:
0.0 )
Features  sex salary are correlated ( chi2 score: 2248.847679013691 , p-value:
0.0 )

Test cannot be performed for features  native_country salary
```

5.0.3 Independent T-test (mean test)

Null hypothesis of T test: the true difference between the real means of two populations is 0.

Conditions to apply the test: - features are independent (true, very small values of correlations) - data of features is approximately normally distributed (all numerical features except `education_num` and `hours_per_week` tend to follow a normal distribution) - data of features has the same variance (not True, but we can set equal_var=False)

According to the independent (two-sample) T-test, all pairs of numerical features do not have (almost) equal population means.

[ ]:
```python
for i in range (0, len(numerical_columns)):
    for j in range(i+1, len(numerical_columns)):
      stat, p = stats.ttest_ind(data[numerical_columns[i]],
                                data[numerical_columns[j]],
                                equal_var=False)
      print('Features ',numerical_columns[i],' ',numerical_columns[j],
            'have  t-value:',stat,' and p-value:',p)
      if p > 0.05:
         print('Accept null hypothesis that the means are equal.')
      else:
         print('Reject the null hypothesis that the means are equal.')
      print()
```

```
Features  age   fnlwgt have  t-value: -396.8377342364698  and p-value: 0.0
Reject the null hypothesis that the means are equal.

Features  age   education_num have  t-value: 452.56451031437916  and p-value:
0.0
Reject the null hypothesis that the means are equal.

Features  age   capital_gain have  t-value: -30.855487077600348  and p-value:
```

```
4.656475534553116e-207
Reject the null hypothesis that the means are equal.

Features  age   capital_loss have  t-value: -26.777975652458114  and p-value:
7.915441431034077e-157
Reject the null hypothesis that the means are equal.

Features  age   hours_per_week have  t-value: -21.272142191900368  and p-value:
3.4944090528664544e-100
Reject the null hypothesis that the means are equal.

Features  fnlwgt   education_num have  t-value: 396.897517755032  and p-value:
0.0
Reject the null hypothesis that the means are equal.

Features  fnlwgt   capital_gain have  t-value: 393.6814417217183  and p-value:
0.0
Reject the null hypothesis that the means are equal.

Features  fnlwgt   capital_loss have  t-value: 396.73259988570044  and p-value:
0.0
Reject the null hypothesis that the means are equal.

Features  fnlwgt   hours_per_week have  t-value: 396.8340122812461  and p-value:
0.0
Reject the null hypothesis that the means are equal.

Features  education_num   capital_gain have  t-value: -31.702695666535618  and
p-value: 2.365101675560344e-218
Reject the null hypothesis that the means are equal.

Features  education_num   capital_loss have  t-value: -42.45753709759881  and
p-value: 0.0
Reject the null hypothesis that the means are equal.

Features  education_num   hours_per_week have  t-value: -529.9070857252188  and
p-value: 0.0
Reject the null hypothesis that the means are equal.

Features  capital_gain   capital_loss have  t-value: 29.363643210652192  and
p-value: 6.792031770799339e-188
Reject the null hypothesis that the means are equal.

Features  capital_gain   hours_per_week have  t-value: 30.802743476016214  and
p-value: 2.298724388726778e-206
Reject the null hypothesis that the means are equal.

Features  capital_loss   hours_per_week have  t-value: 25.80580371095441  and
```

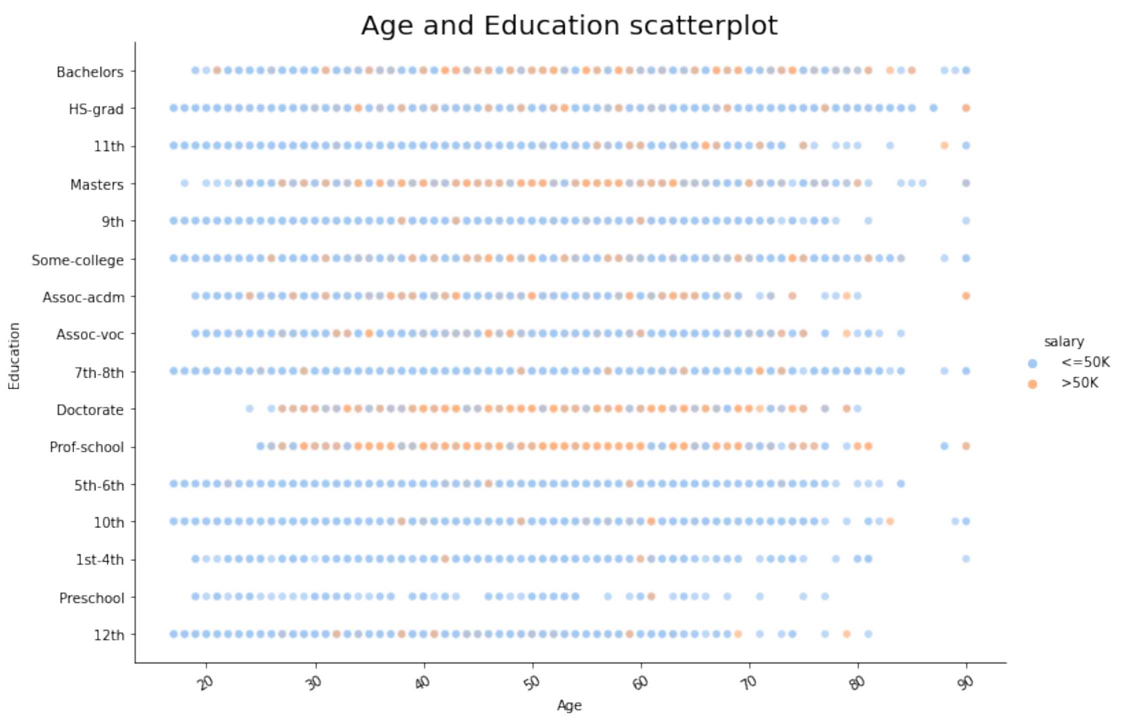
```
p-value: 7.256522197714283e-146
Reject the null hypothesis that the means are equal.
```

5.0.4 Scatterplots for pairs of features

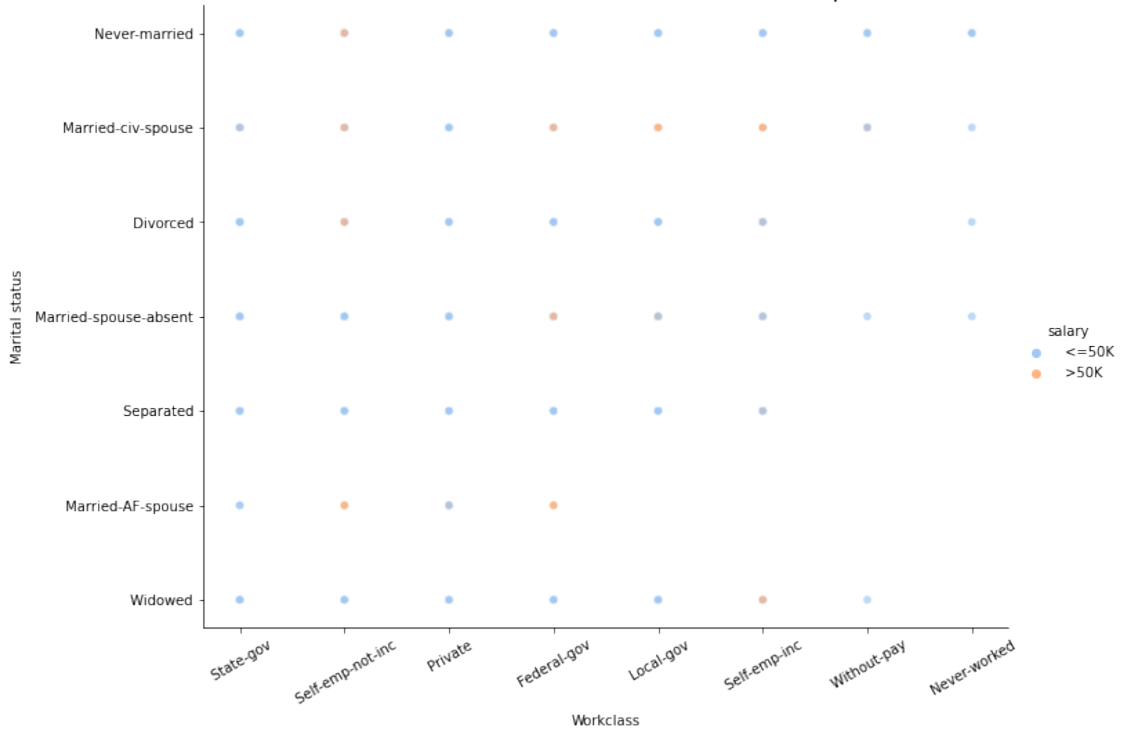
Interpretations for (some of the) scatterplots: - `age` and `education`: We can observe that persons earning >50K usually have Doctorate, Professional School, Masters or Bachelor's education level. The common start age of persons with Doctorate and Professional School education level is usually 25.

- `workclass` and `marital_status`: We can observe that persons earning >50K are usually married and are either self-employed or work for the Government.
- `education` and `capital_gain`: The highest levels of capital gains were attained by persons who have Bachelors, Masters or Preschool education levels. However, in general, higher gains are attained by people who at least completed highschool. Persons who earn >50K usually have higher capital gains.
- `race` and `sex`: White male usually gain >50K

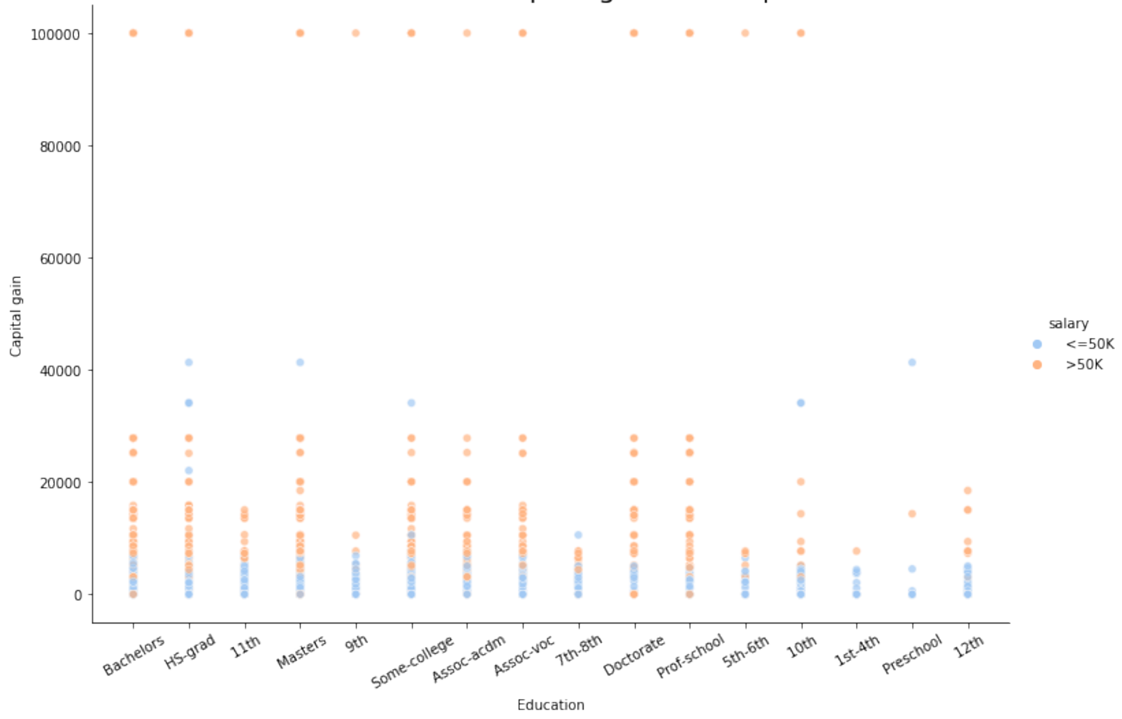
[ ]:
```
#all scatterplots
# for idx in range(0,len(data.columns)):
#    if data.columns[idx]!='salary':
#     for idy in range(idx+1, len(data.columns)):
#        if data.columns[idy]!='salary':

#           title = str(data.columns[idx]).capitalize().replace("_", " ")
#           title+= " and "
#           title += str(data.columns[idy]).capitalize().replace("_", " ")
#           title += " scatterplot"


#           p = sns.relplot(x=data.columns[idx], y=data.columns[idy],
#                           alpha=.7, palette="pastel", hue= 'salary',
#                           data=data, height=7, aspect=1.5 )

#           plt.xticks(rotation=30)
#           plt.xlabel(str(data.columns[idx]).capitalize().replace("_", " "))
#           plt.ylabel(str(data.columns[idy]).capitalize().replace("_", " "))
#           plt.title(title, fontsize=20)

#           plt.show()

#only selected scatterplots
col_pairs = [('age', 'education'),('workclass', 'marital_status'),
             ('education', 'capital_gain'),('race', 'sex')]
for (col1, col2) in col_pairs:
          title = col1.capitalize().replace("_", " ")
          title+= " and "
          title += col2.capitalize().replace("_", " ")
          title += " scatterplot"


          p = sns.relplot(x=data[col1], y=data[col2],
                          alpha=.7, palette="pastel", hue= 'salary',
                          data=data, height=7, aspect=1.5 )

          plt.xticks(rotation=30)
          plt.xlabel(str(col1).capitalize().replace("_", " "))
          plt.ylabel(str(col2).capitalize().replace("_", " "))
          plt.title(title, fontsize=20)

          plt.show()
```

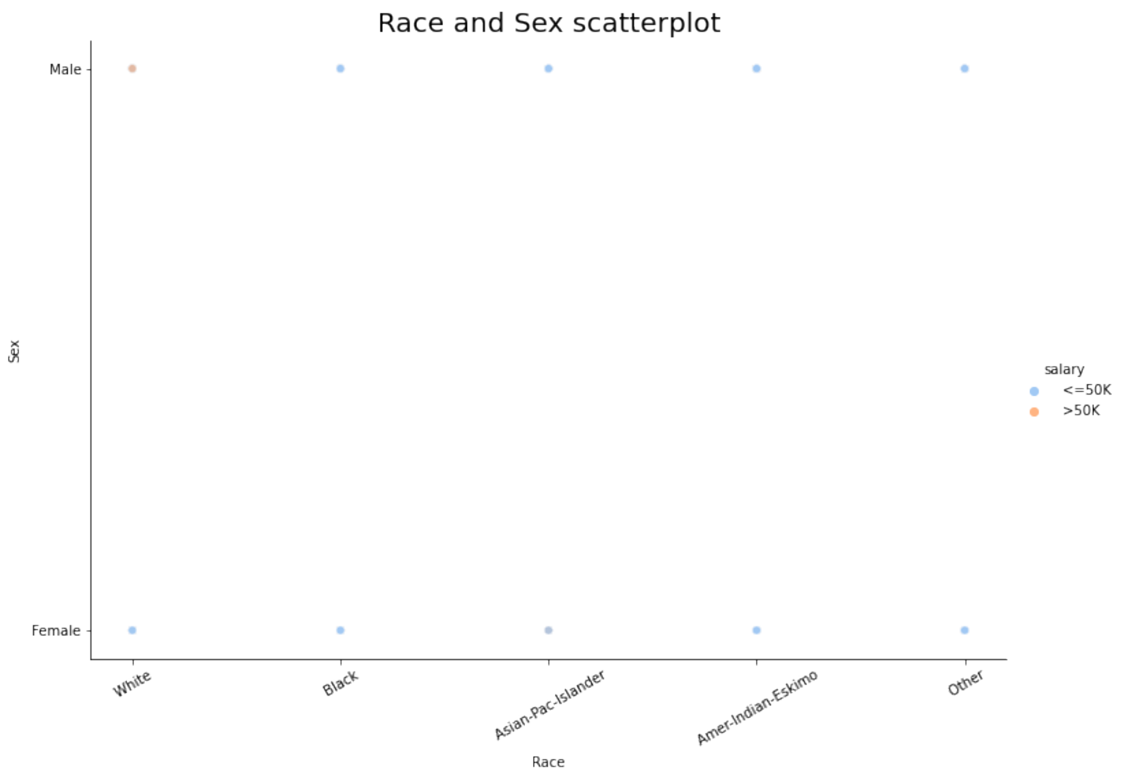


Workclass and Marital status scatterplot



Education and Capital gain scatterplot





5.0.5 3D scatterplots

Interpretations:

- `workclass`, `salary`, `occupation` 3D plot: An unusual thing to observe is the fact that persons earning >50K who are categorized by `workclass` as "Without pay", have occupations such as Handlers-Cleaners and Machine Operator Inspectors. Another thing to notice is that except 'Armed Forces' and 'Private house services', all other occupations are present among all workclasses.
- `salary`, `marital_status`, `education` 3D plot: It can be observed that people earning >50K usually have a higher level of education that 6th grade. The marital status of people earning >50K who have an attained level of education <=6th grade, their marital status is either: never married, married with civil spouse or separated.
- `race`, `salary`, `capital_gain` 3D plot: It can be observed that there are very few old people (above 70) who work: more than 55 hours per week (in case of persons earning >50K) and more than 65 hours per week (in case of persons earning <=50K).

[ ]:
```
#all 3D scatterplots
# for comb in combinations(data.columns, 3):
#   comb_l= list(comb)
```