EDA with Pandas in Banking

January 2, 2022

EDA with Pandas in Banking

Import the necessary libraries and ignore warnings

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
import warnings
warnings.filterwarnings('ignore')
```

Dataset Exploration In this section you will explore the sourse dataset.

Let's read the data and look at the first 5 rows using the head method. The number of the output rows from the dataset is determined by the head method parameter.

[4]:
```
df = pd.read_csv('bank-additional-full.csv', sep = ';')
df.head(5)
```

[4]:

	age	job	marital	education	default	housing	loan	contact	\
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	

	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate	\
0	may	mon	…	1	999	0	nonexistent	1.1	
1	may	mon	…	1	999	0	nonexistent	1.1	
2	may	mon	…	1	999	0	nonexistent	1.1	
3	may	mon	…	1	999	0	nonexistent	1.1	
4	may	mon	…	1	999	0	nonexistent	1.1	

	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	93.994	-36.4	4.857	5191.0	no
1	93.994	-36.4	4.857	5191.0	no
2	93.994	-36.4	4.857	5191.0	no
3	93.994	-36.4	4.857	5191.0	no
4	93.994	-36.4	4.857	5191.0	no

```
[5 rows x 21 columns]
```

Let's look at the dataset size, feature names and their types

[6]:
```
df.shape
# The dataset contains 41188 objects (rows),
#for each of which 21 features are set (columns), including 1 target feature (y).
```

[6]:
```
(41188, 21)
```

[7]:
```
# Attributing information
df.columns
```

[7]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

[8]:
```
#To see the general information on all the DataFrame features (columns), we use the info method:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
```

```
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

[9]:
```
df.describe()
```

[9]:

	age	duration	campaign	pdays	previous
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963
std	10.42125	259.279249	2.770014	186.910907	0.494901
min	17.00000	0.000000	1.000000	0.000000	0.000000
25%	32.00000	102.000000	1.000000	999.000000	0.000000
50%	38.00000	180.000000	2.000000	999.000000	0.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000
max	98.00000	4918.000000	56.000000	999.000000	7.000000

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	1.570960	0.578840	4.628198	1.734447	72.251528
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

[10]:
```
df.describe(include = ["object"])
```

[10]:

	job	marital	education	default	housing	loan	contact
count	41188	41188	41188	41188	41188	41188	41188
unique	12	4	8	3	3	3	2
top	admin.	married	university.degree	no	yes	no	cellular
freq	10422	24928	12168	32588	21576	33950	26144

	month	day_of_week	poutcome	y
count	41188	41188	41188	41188
unique	10	5	3	2
top	may	thu	nonexistent	no
freq	13769	8623	35563	36548

df.describe(include = [“object”]) df.describe(include = [“object”]) The result shows that the average client refers to administrative staff (job = admin.), is married (marital = married) and has a university degree (education = university.degree).

For categorical (type object) and boolean (type bool) features you can use the value_counts method. Let’s look at the target feature (y) distribution:

[11]:
```
df["y"].value_counts()
```

[11]:
```
no     36548
yes     4640
Name: y, dtype: int64
```

[13]:
```
df["marital"].value_counts() ##YOUR CODE GOES HERE##
```

[13]:
```
married     24928
single      11568
divorced     4612
unknown        80
Name: marital, dtype: int64
```

Sorting A DataFrame can be sorted by a few feature values. In our case, for example, by duration (ascending = False for sorting in descending order):

[15]:
```
df.sort_values(by = "duration", ascending = False).head()
```

[15]:

	age	job	marital	education	default	housing	loan	\
24091	33	technician	single	professional.course	no	yes	no	
22192	52	blue-collar	married	basic.4y	no	no	no	
40537	27	admin.	single	high.school	no	no	no	
13820	31	technician	married	professional.course	no	no	no	
7727	37	unemployed	married	professional.course	no	yes	no	

	contact	month	day_of_week	…	campaign	pdays	previous	\
24091	telephone	nov	mon	…	1	999	0	
22192	telephone	aug	thu	…	3	999	0	
40537	telephone	aug	fri	…	1	999	0	
13820	cellular	jul	thu	…	1	999	0	
7727	telephone	may	fri	…	2	999	0	

	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	\
24091	nonexistent	-0.1	93.200	-42.0	4.406	
22192	nonexistent	1.4	93.444	-36.1	4.963	
40537	nonexistent	-1.7	94.027	-38.3	0.888	
13820	nonexistent	1.4	93.918	-42.7	4.963	
7727	nonexistent	1.1	93.994	-36.4	4.864	

	nr.employed	y
24091	5195.8	no
22192	5228.1	yes
40537	4991.6	no
13820	5228.1	yes
7727	5191.0	yes

```
[5 rows x 21 columns]
```

[16]:
```
df.sort_values(by = ["age", "duration"], ascending = [True, False]).head()
```

[16]:
```
       age      job marital education default  housing     loan    contact  \
38274   17  student  single   unknown      no       no      yes  cellular
37579   17  student  single  basic.9y      no  unknown  unknown  cellular
37140   17  student  single   unknown      no      yes       no  cellular
37539   17  student  single  basic.9y      no      yes       no  cellular
37558   17  student  single  basic.9y      no      yes       no  cellular

      month day_of_week  …  campaign  pdays  previous  poutcome  \
38274   oct         tue  …         1      2         2   success
37579   aug         fri  …         2    999         1   failure
37140   aug         wed  …         3      4         2   success
37539   aug         fri  …         2    999         2   failure
37558   aug         fri  …         3      4         2   success

      emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed    y
38274         -3.4          92.431          -26.9      0.742       5017.5  yes
37579         -2.9          92.201          -31.4      0.869       5076.2  yes
37140         -2.9          92.201          -31.4      0.884       5076.2   no
37539         -2.9          92.201          -31.4      0.869       5076.2   no
37558         -2.9          92.201          -31.4      0.869       5076.2   no

[5 rows x 21 columns]
```

Application of functions: apply, map etc. Apply the function to each column:

[17]:
```
df.apply(np.max) # age =98, oldest
```

[17]:
```
age                  98
job             unknown
marital         unknown
education       unknown
default             yes
housing             yes
loan                yes
contact       telephone
month               sep
day_of_week         wed
duration           4918
campaign             56
pdays               999
previous              7
poutcome        success
emp.var.rate        1.4
```

```
cons.price.idx       94.767
cons.conf.idx         -26.9
euribor3m             5.045
nr.employed          5228.1
y                       yes
dtype: object
```

Apply the function to each column cell

The map can also be used for the values replacement in a column by passing it as an argument dictionary in form of {old_value: new_value}

[19]:
```
d = {"no": 0, "yes": 1}
df["y"] = df["y"].map(d)
df.head()
```

[19]:

	age	job	marital	education	default	housing	loan	contact	\
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	

	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate	\
0	may	mon	…	1	999	0	nonexistent	1.1	
1	may	mon	…	1	999	0	nonexistent	1.1	
2	may	mon	…	1	999	0	nonexistent	1.1	
3	may	mon	…	1	999	0	nonexistent	1.1	
4	may	mon	…	1	999	0	nonexistent	1.1	

	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	93.994	-36.4	4.857	5191.0	0
1	93.994	-36.4	4.857	5191.0	0
2	93.994	-36.4	4.857	5191.0	0
3	93.994	-36.4	4.857	5191.0	0
4	93.994	-36.4	4.857	5191.0	0

[5 rows x 21 columns]

[20]:
```
print("Share of attracted clients =", '{:.1%}'.format(df["y"].mean()))
```

Share of attracted clients = 11.3%

[21]:
```
df[df["y"] == 1].mean()
```

[21]:
```
age                40.913147
duration          553.191164
campaign            2.051724
```

```
pdays              792.035560
previous             0.492672
emp.var.rate        -1.233448
cons.price.idx      93.354386
cons.conf.idx      -39.789784
euribor3m            2.123135
nr.employed       5095.115991
y                    1.000000
dtype: float64
```

What is the average call duration for the attracted clients?

[23]:
```
acd = round(df[df["y"] == 1]["duration"].mean(), 2)
acd_in_min = acd // 60
print("Average call duration for attracted clients =", acd_in_min, "min",␣
 ↪int(acd) % 60, "sec")
```

```
Average call duration for attracted clients = 9.0 min 13 sec
```

What is the average age of attracted (y == 1) and unmarried ('marital' == 'single') clients?

[24]:
```
print("Average age of attracted clients =", int(df[(df["y"] == 1) &␣
 ↪(df["marital"] == "single")]["age"].mean()), "years")
```

```
Average age of attracted clients = 31 years
```

Pivot tables

[25]:
```
pd.crosstab(df["y"], df["marital"])
```

[25]:
```
marital  divorced  married  single  unknown
y
0            4136    22396    9948       68
1             476     2532    1620       12
```

[26]:
```
pd.crosstab(df["y"],
            df["marital"],
            normalize = 'index')
```

[26]:
```
marital  divorced   married    single   unknown
y
0        0.113166  0.612783  0.272190  0.001861
1        0.102586  0.545690  0.349138  0.002586
```

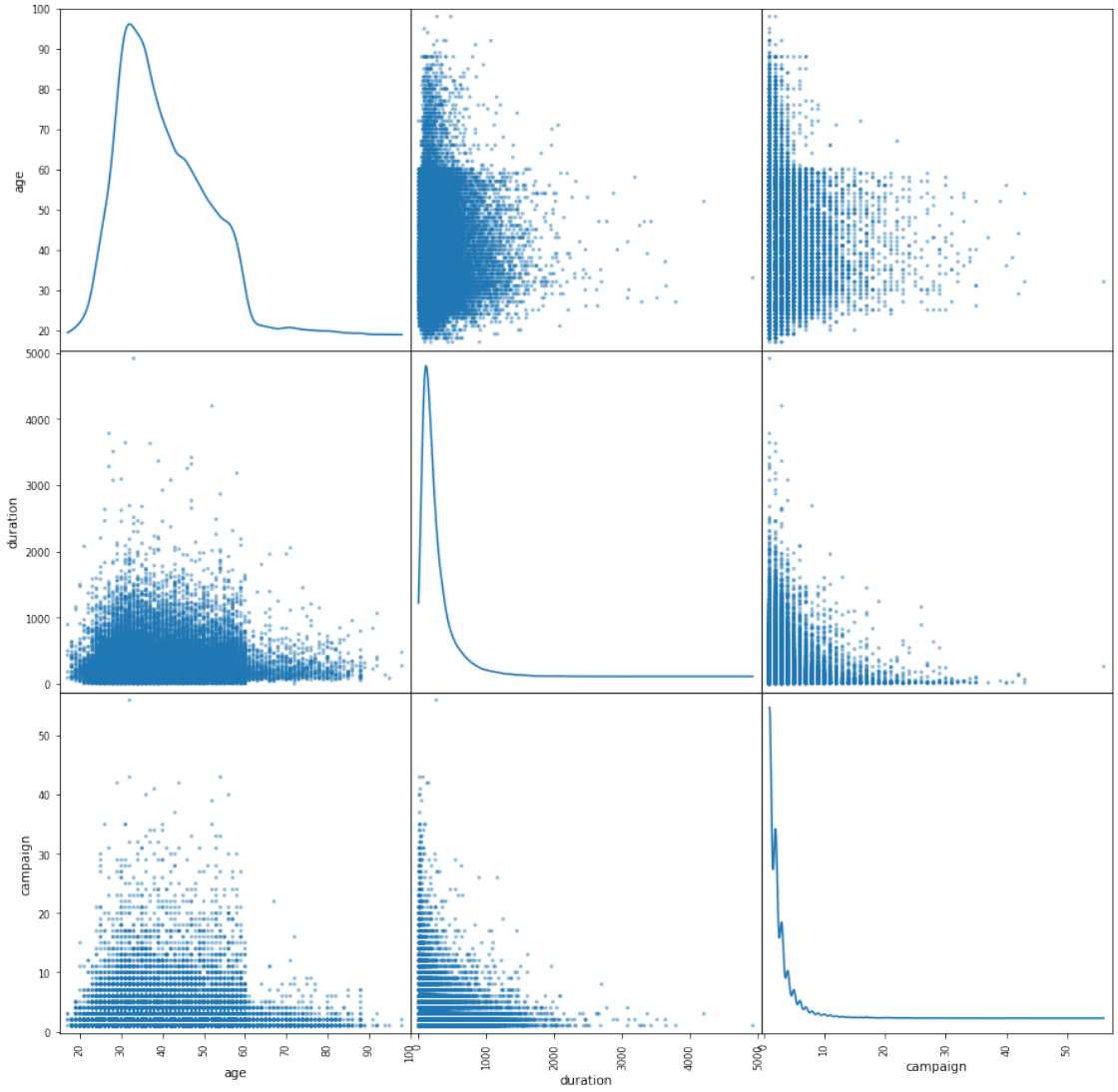
[27]:
```
df.pivot_table(
    ["age", "duration"],
    ["job"],
    aggfunc = "mean",
).head(10)
```

[27]:

job	age	duration
admin.	38.187296	254.312128
blue-collar	39.555760	264.542360
entrepreneur	41.723214	263.267857
housemaid	45.500000	250.454717
management	42.362859	257.058140
retired	62.027326	273.712209
self-employed	39.949331	264.142153
services	37.926430	258.398085
student	25.894857	283.683429
technician	38.507638	250.232241

Visualization in Pandas

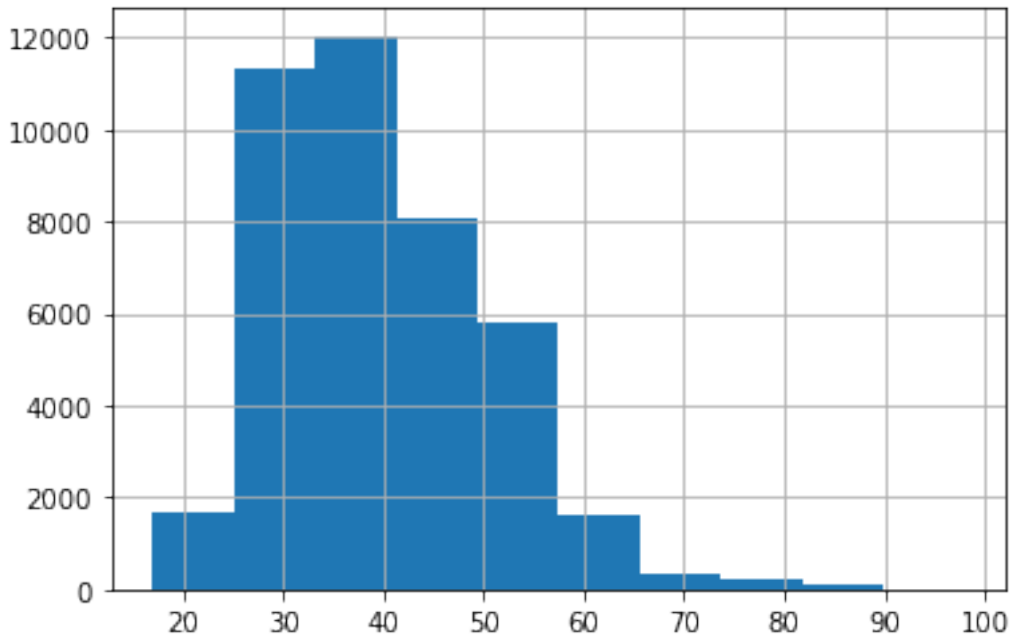
[28]:
```
pd.plotting.scatter_matrix(
    df[["age", "duration", "campaign"]],
    figsize = (15, 15),
    diagonal = "kde")
plt.show()
```



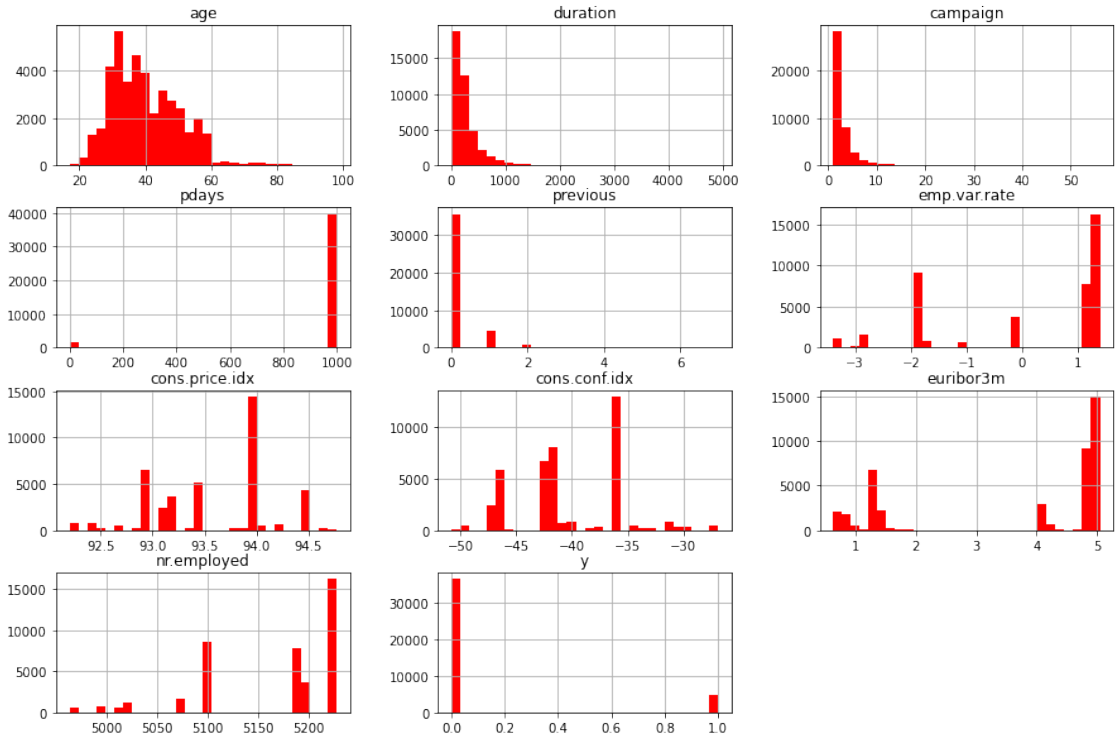
We can build a separate histogram for each feature:

[29]:
```
df["age"].hist()
```

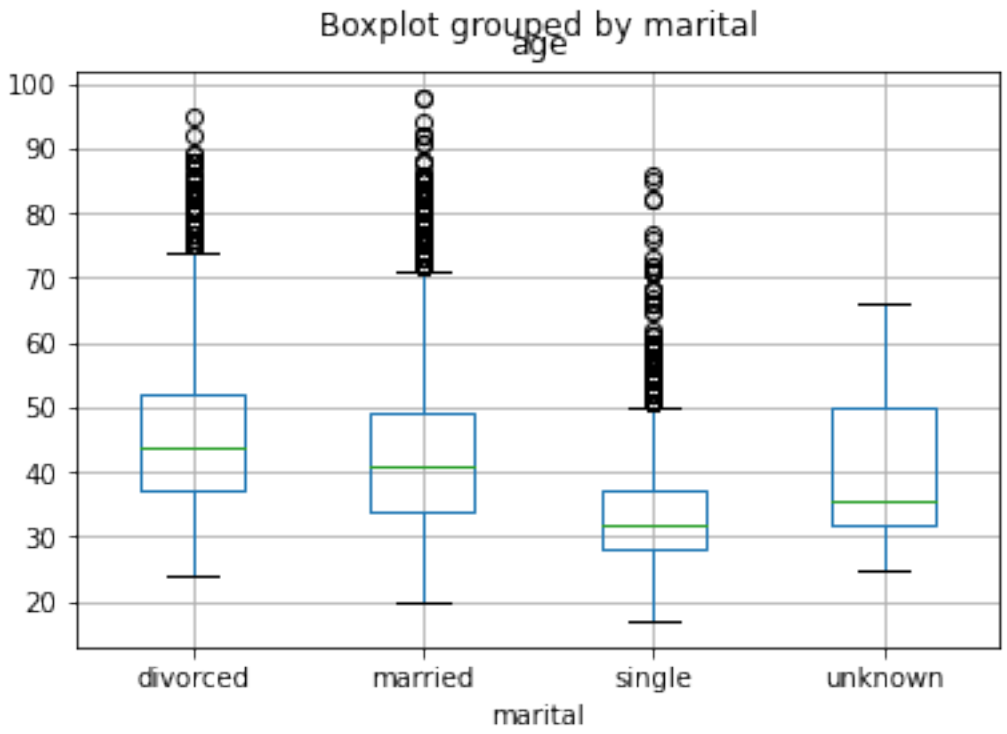
[29]: <AxesSubplot:>



[30]:
```
df.hist(color = "r",
        bins = 30,
        figsize = (15, 10))
plt.show()
```



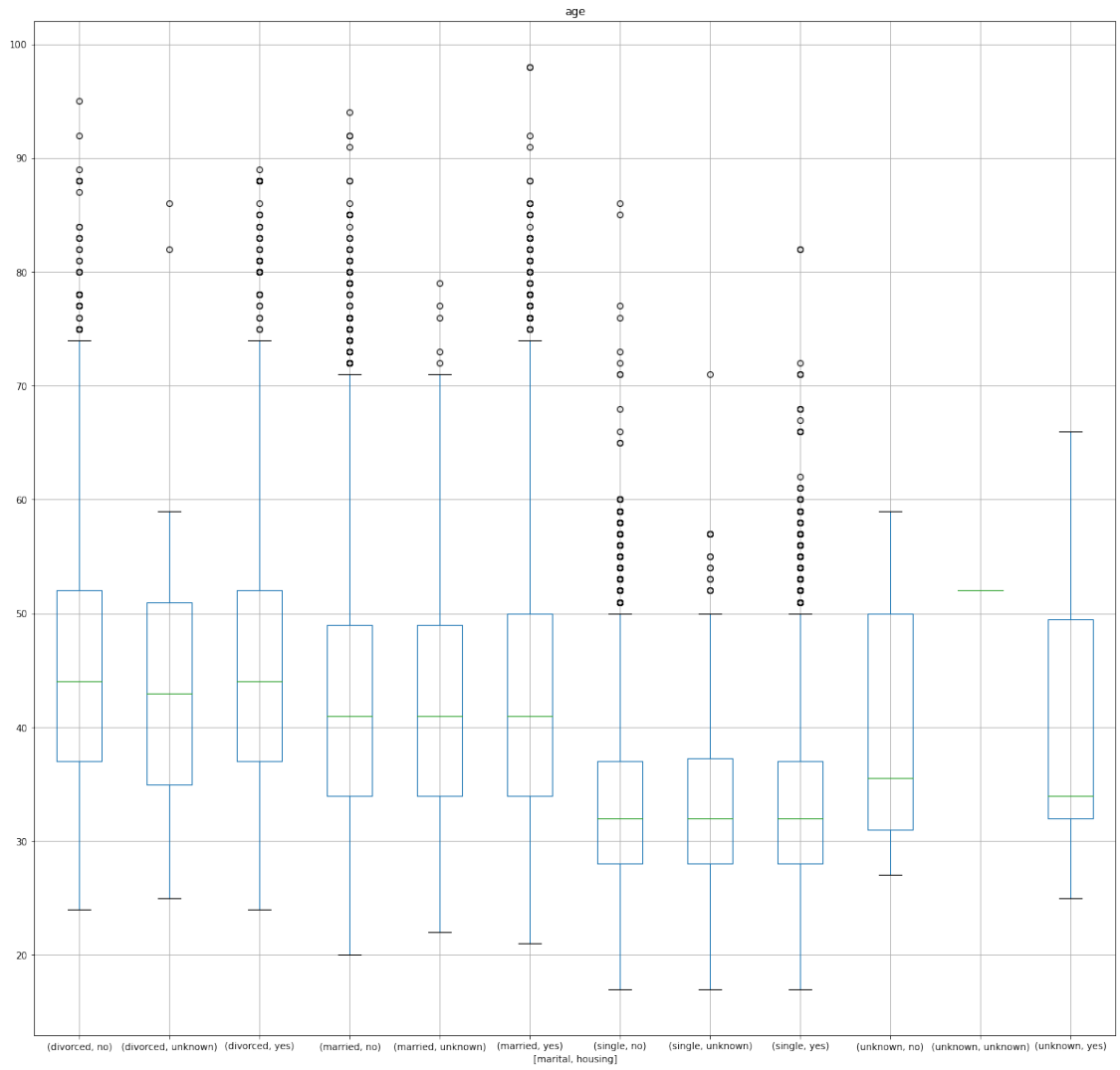
[31]:
```
df.boxplot(column = "age",
           by = "marital")
plt.show()
```



data grouping

[32]:
```
df.boxplot(column = "age",
           by = ["marital", "housing"],
           figsize = (20, 20))
```

[32]:
```
<AxesSubplot:title={'center':'age'}, xlabel='[marital, housing]'>
```



Question 1 List 10 clients with the largest number of contacts.

[ ]:
```
Question 1
List 10 clients with the largest number of contacts.
```

[33]:
```
df.sort_values(by = "campaign", ascending = False).head(10)
```

[33]:

	age	job	marital	education	default	housing	\
4107	32	admin.	married	university.degree	unknown	unknown	
18728	54	admin.	married	university.degree	unknown	yes	
13447	32	technician	single	university.degree	no	yes	
4168	29	technician	married	professional.course	no	yes	

```
5304    44        retired  married            basic.9y       no      yes
11033   38    blue-collar  married            basic.4y       no      yes
18754   36         admin.   single  university.degree       no       no
11769   56  self-employed  married  professional.course       no       no
4114    52   entrepreneur  married  university.degree       no       no
11593   43     technician  married        high.school       no      yes

          loan    contact month day_of_week  …  campaign  pdays  previous  \
4107   unknown  telephone   may         mon  …        56    999         0
18728       no   cellular   jul         thu  …        43    999         0
13447      yes  telephone   jul         wed  …        43    999         0
4168        no  telephone   may         mon  …        42    999         0
5304        no  telephone   may         fri  …        42    999         0
11033       no  telephone   jun         wed  …        41    999         0
18754       no   cellular   jul         thu  …        40    999         0
11769      yes  telephone   jun         fri  …        40    999         0
4114        no  telephone   may         mon  …        39    999         0
11593       no  telephone   jun         fri  …        37    999         0

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
4107   nonexistent          1.1          93.994          -36.4      4.858
18728  nonexistent          1.4          93.918          -42.7      4.968
13447  nonexistent          1.4          93.918          -42.7      4.962
4168   nonexistent          1.1          93.994          -36.4      4.858
5304   nonexistent          1.1          93.994          -36.4      4.857
11033  nonexistent          1.4          94.465          -41.8      4.962
18754  nonexistent          1.4          93.918          -42.7      4.968
11769  nonexistent          1.4          94.465          -41.8      4.959
4114   nonexistent          1.1          93.994          -36.4      4.858
11593  nonexistent          1.4          94.465          -41.8      4.959

       nr.employed  y
4107        5191.0  0
18728       5228.1  0
13447       5228.1  0
4168        5191.0  0
5304        5191.0  0
11033       5228.1  0
18754       5228.1  0
11769       5228.1  0
4114        5191.0  0
11593       5228.1  0

[10 rows x 21 columns]
```

Question 2 Determine the median age and the number of contacts for different levels of client education.

[34]:
```
df.pivot_table(
    ["age", "campaign"],
    ["education"],
    aggfunc = ["mean", "count"],
)
```

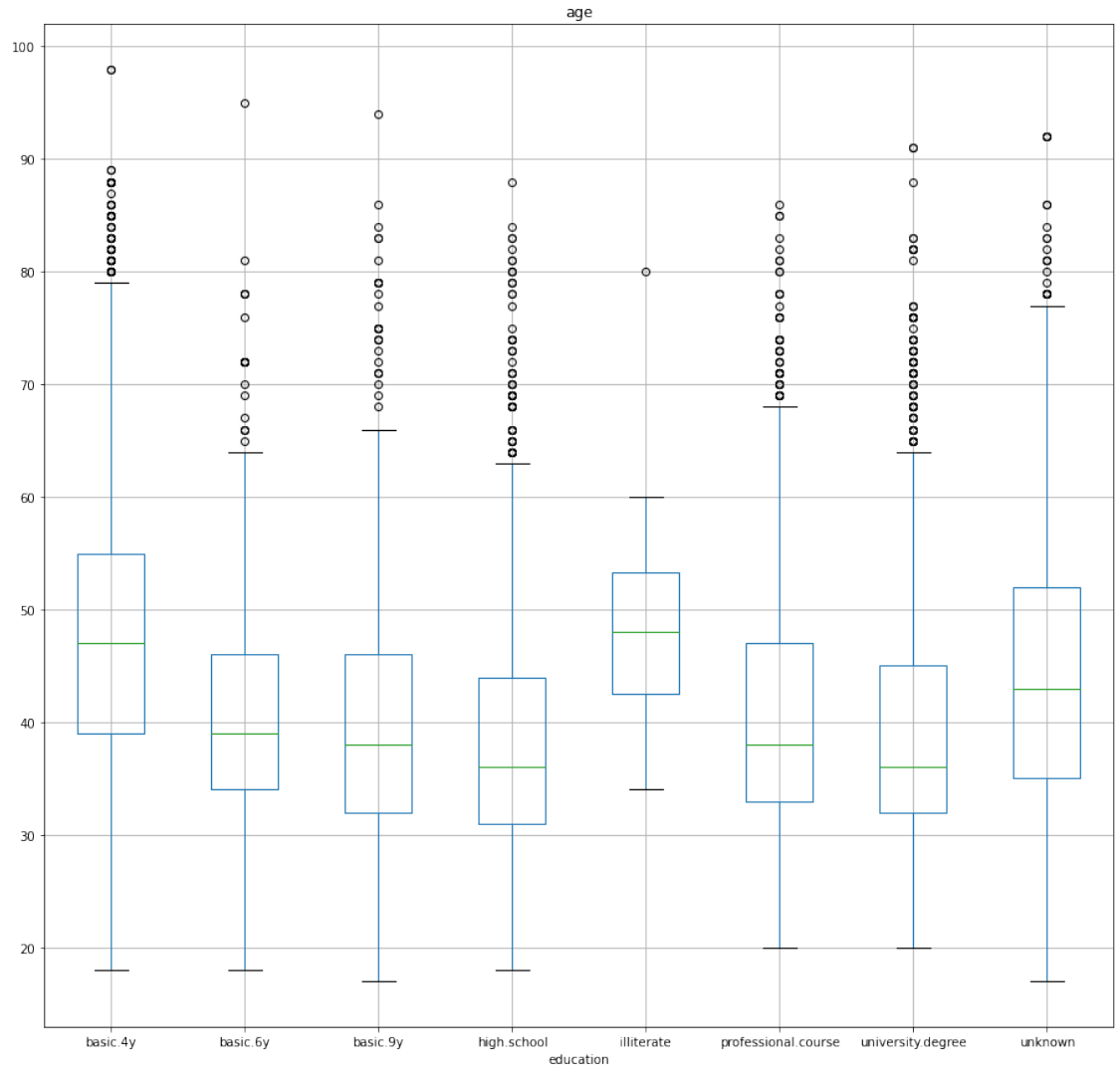
[34]:

	mean		count	
	age	campaign	age	campaign
education				
basic.4y	47.596504	2.600575	4176	4176
basic.6y	40.448953	2.556283	2292	2292
basic.9y	39.061208	2.532341	6045	6045
high.school	37.998213	2.568576	9515	9515
illiterate	48.500000	2.277778	18	18
professional.course	40.080107	2.586115	5243	5243
university.degree	38.879191	2.563527	12168	12168
unknown	43.481225	2.596187	1731	1731

Question 3 Output box plot to analyze the client age distribution by their education level.

[35]:
```
df.boxplot(column = "age",
  by = "education",
  figsize = (15, 15))
```

[35]:
```
<AxesSubplot:title={'center':'age'}, xlabel='education'>
```


Boxplot grouped by education
age
100
90
80
70
60
50
40
30
20
basic.4y
basic.6y
basic.9y
high.school
illiterate
professional.course
university.degree
unknown
education

[ ]:

[ ]:

[ ]:

[ ]: