Banking Data Analysis

Samarth Chetan

MS in Data Analytics Engineering

Northeastern University

In [1]:
```
# importing necessary libraries for conducting EDA
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
plt.rcParams["figure.figsize"] = (8, 6)

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# reading the dataset and taking a look at the first five rows of data
df = pd.read_csv('bank-additional.csv', sep = ';')
df.head(5)
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	pout
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	none
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	none
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	none
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	none
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	none

5 rows × 21 columns

In [4]:
```
# playing around with the dataset - exploring features
df.shape
df.tail(5)
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcc
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	jul	thu	...	1	999	0	nonexis
4115	39	admin.	married	high.school	no	yes	no	telephone	jul	fri	...	1	999	0	nonexis
4116	27	student	single	high.school	no	no	no	cellular	may	mon	...	2	999	1	fai
4117	58	admin.	married	high.school	no	no	no	cellular	aug	fri	...	1	999	0	nonexis
4118	34	management	single	high.school	no	yes	no	cellular	nov	wed	...	1	999	0	nonexis

5 rows × 21 columns

In [5]:
```
# what are the columns involved in the dataset
df.columns
```

Out[5]:
```
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')
```

Input features (column names):

1. `age` - client's age in years (numeric)
2. `job` - type of job (categorical: `admin.`, `blue-collar`, `entrepreneur`, `housemaid`, `management`, `retired`, `self-employed`, `services`, `student`, `technician`, `unemployed`, `unknown`)
3. `marital` - marital status (categorical: `divorced`, `married`, `single`, `unknown`)
4. `education` - client's education (categorical: `basic.4y`, `basic.6y`, `basic.9y`, `high.school`, `illiterate`, `professional.course`, `university.degree`, `unknown`)
5. `default` - has credit in default? (categorical: `no`, `yes`, `unknown`)
6. `housing` - has housing loan? (categorical: `no`, `yes`, `unknown`)
7. `loan` - has personal loan? (categorical: `no`, `yes`, `unknown`)
8. `contact` - contact communication type (categorical: `cellular`, `telephone`)
9. `month` - last contact month of the year (categorical: `jan`, `feb`, `mar`, ..., `nov`, `dec`)

10. `day_of_week` - last contact day of the week (categorical: `mon`, `tue`, `wed`, `thu`, `fri`)
11. `duration` - last contact duration, in seconds (numeric).
12. `campaign` - number of contacts performed and for this client during this campaign (numeric, includes the last contact)
13. `pdays` - number of days that have passed after the client was last contacted from the previous campaign (numeric; 999 means the client has not been previously contacted)
14. `previous` - number of contacts performed for this client before this campaign (numeric)
15. `poutcome` - outcome of the previous marketing campaign (categorical: `failure`, `nonexistent`, `success`)
16. `emp.var.rate` - employment variation rate, quarterly indicator (numeric)
17. `cons.price.idx` - consumer price index, monthly indicator (numeric)
18. `cons.conf.idx` - consumer confidence index, monthly indicator (numeric)
19. `euribor3m` - euribor 3 month rate, daily indicator (numeric)
20. `nr.employed` - number of employees, quarterly indicator (numeric)

Output feature (desired target):

1. `y` - has the client subscribed a term deposit? (binary: `yes`, `no`)

In [6]:
```
# general information related to data
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             4119 non-null   int64  
 1   job             4119 non-null   object 
 2   marital         4119 non-null   object 
 3   education       4119 non-null   object 
 4   default         4119 non-null   object 
 5   housing         4119 non-null   object 
 6   loan            4119 non-null   object 
 7   contact         4119 non-null   object 
 8   month           4119 non-null   object 
 9   day_of_week     4119 non-null   object 
 10  duration        4119 non-null   int64  
 11  campaign        4119 non-null   int64  
 12  pdays           4119 non-null   int64  
 13  previous        4119 non-null   int64  
 14  poutcome        4119 non-null   object 
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object 
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
None
```

In [7]:
```
df.describe()
```

Out[7]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356	5166.481695
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591	73.667904
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000	4963.600000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000	5099.100000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

In [8]:
```
df.describe(include = ["object"])
```

Out[8]:

	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	y
count	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	1012	2509	1264	3315	2175	3349	2652	1378	860	3523	3668

In [9]:
```
df["y"].value_counts()
```

Out[9]:
```
no     3668
yes     451
Name: y, dtype: int64
```

4640 clients (11.3%) of 41188 issued a term deposit

In [10]:
```
df["marital"].value_counts(normalize = True)
```

Out[10]:
```
married     0.609128
single      0.279922
divorced    0.108279
unknown     0.002671
Name: marital, dtype: float64
```

61% (0.61) of clients are married which has a great significance while conducting marketing campaigns

In [11]:
```
# sorting the dataframe
df.sort_values(by = "duration", ascending = False).head()
```

Out[11]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	p
2231	31	technician	married	professional.course	no	no	no	cellular	jul	thu	...	1	999	0	no
1546	46	admin.	divorced	high.school	no	yes	no	telephone	oct	fri	...	1	999	0	no
1392	47	admin.	divorced	university.degree	no	yes	no	telephone	jun	fri	...	3	999	0	no
1685	33	blue-collar	single	high.school	no	no	no	cellular	may	mon	...	1	999	0	no
3266	49	blue-collar	married	basic.6y	no	yes	no	telephone	may	fri	...	2	999	0	no

5 rows × 21 columns

In [12]:
```
df.sort_values(by = ["age", "duration"], ascending = [True, False]).head()
```

Out[12]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
477	18	student	single	unknown	no	no	no	cellular	sep	thu	...	1	3	1	success
899	18	student	single	unknown	no	yes	yes	telephone	aug	wed	...	1	999	0	nonexistent
1661	18	student	single	unknown	no	yes	no	cellular	may	thu	...	1	7	2	success
1887	19	student	single	high.school	unknown	yes	no	cellular	may	tue	...	4	999	0	nonexistent
3268	20	blue-collar	single	high.school	no	yes	no	cellular	may	wed	...	1	999	0	nonexistent

5 rows × 21 columns

In [13]:
```
# applying the function to every column
df.apply(np.max)
```

Out[13]:
```
age                 88
job            unknown
marital        unknown
education      unknown
default            yes
housing            yes
loan               yes
contact      telephone
month              sep
day_of_week        wed
duration          3643
campaign            35
pdays              999
previous             6
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
```

Age of the oldest client - 98 years Number of contacts reached(campaign) - 56

In [14]:
```
d = {"no": 0, "yes": 1}
df["y"] = df["y"].map(d)
df.head()
```

Out[14]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	pout
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	none
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	none
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	none
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	none
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	none

5 rows × 21 columns

In [16]:
```
# extracting data from the dataset
print("Attracted client share =", '{:.1%}'.format(df["y"].mean()))
```

```
Attracted client share = 10.9%
```

In [17]:
```
# finding the mean value
df[df["y"] == 1].mean()
```

Out[17]:
```
age                41.889135
duration          560.787140
campaign            1.980044
pdays             778.722838
previous            0.585366
emp.var.rate       -1.177384
cons.price.idx     93.417268
cons.conf.idx     -39.786475
euribor3m           2.145448
nr.employed       5093.118625
y                   1.000000
dtype: float64
```

Attracted clients average age - 40 years Average number of calls required to attract them - 2

In [19]:
```
# calculating the average call duration of the attracted client
acd = round(df[df["y"] == 1]["duration"].mean(), 2)
acd_in_min = acd // 60
print("Attracted client average call duration =", acd_in_min, "min", int(acd) % 60, "sec")
```

```
Attracted client average call duration = 9.0 min 20 sec
```

In [21]:
```
# finding the average age and marital status of attracted clients who are single
print("Average age of attracted clients =", int(df[(df["y"] == 1) & (df["marital"] == "single")]["age"].mean()), "years")
```

```
Average age of attracted clients = 32 years
```

In [22]:
```
df[-1:]
```

Out[22]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcom
4118	34	management	single	high.school	no	yes	no	cellular	nov	wed	...	1	999	0	nonexister

1 rows × 21 columns

In [23]:
```
# cross-tabulation
pd.crosstab(df["y"], df["marital"])
```

Out[23]:

marital	divorced	married	single	unknown
y				
0	403	2257	998	10
1	43	252	155	1

In [25]:
```
# how many clients are married but have not yet issued a deposit
pd.crosstab(df["y"],
            df["marital"],
            normalize = 'index')
```

Out[25]:

marital	divorced	married	single	unknown
y				
0	0.109869	0.615322	0.272083	0.002726
1	0.095344	0.558758	0.343681	0.002217

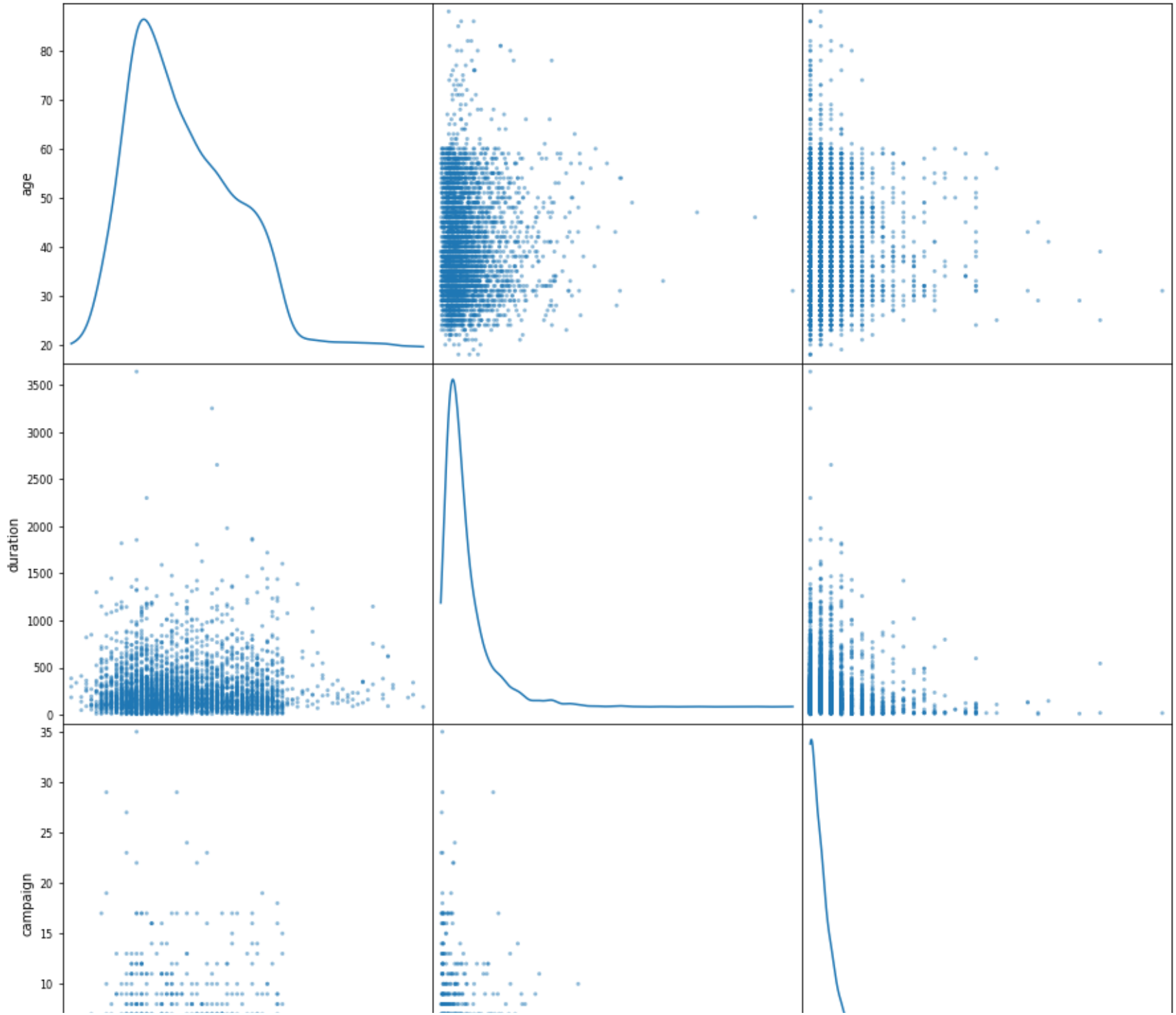
In [28]:
```
# pivot tables - can be utilized for various tasks but mainly used for marketing campaigns
df.pivot_table(
    ["age", "duration"],
    ["job"],
    aggfunc = "mean",
).head(10)
```

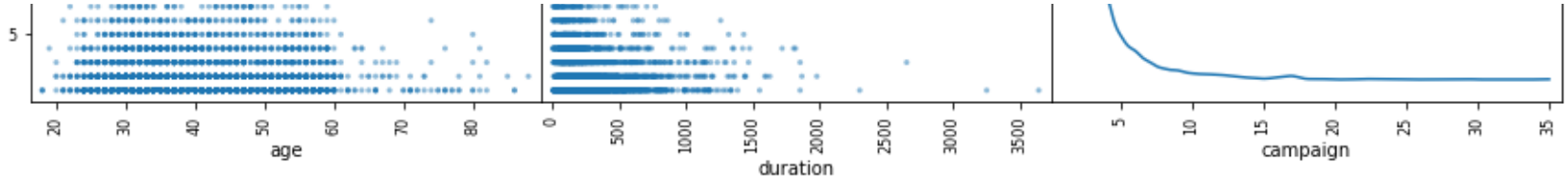
Out[28]:

job	age	duration
admin.	38.240119	261.871542
blue-collar	39.265837	261.852941
entrepreneur	42.202703	249.202703
housemaid	45.672727	229.663636
management	42.429012	246.799383
retired	60.873494	311.789157
self-employed	40.679245	254.924528
services	38.513995	232.529262
student	26.695122	287.134146
technician	38.622287	253.286541

In [29]:

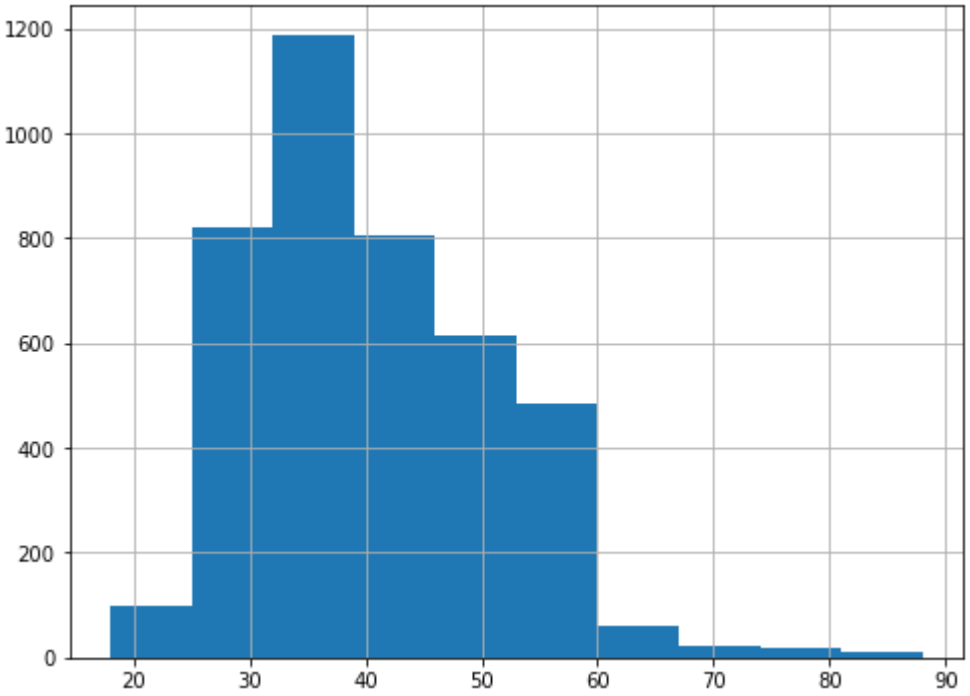
```
# creating a pandas scatter plot
pd.plotting.scatter_matrix(
    df[["age", "duration", "campaign"]],
    figsize = (15, 15),
    diagonal = "kde")
plt.show()
```


age
duration
campaign
80
70
60
50
40
30
20
3500
3000
2500
2000
1500
1000
500
0
35
30
25
20
15
10



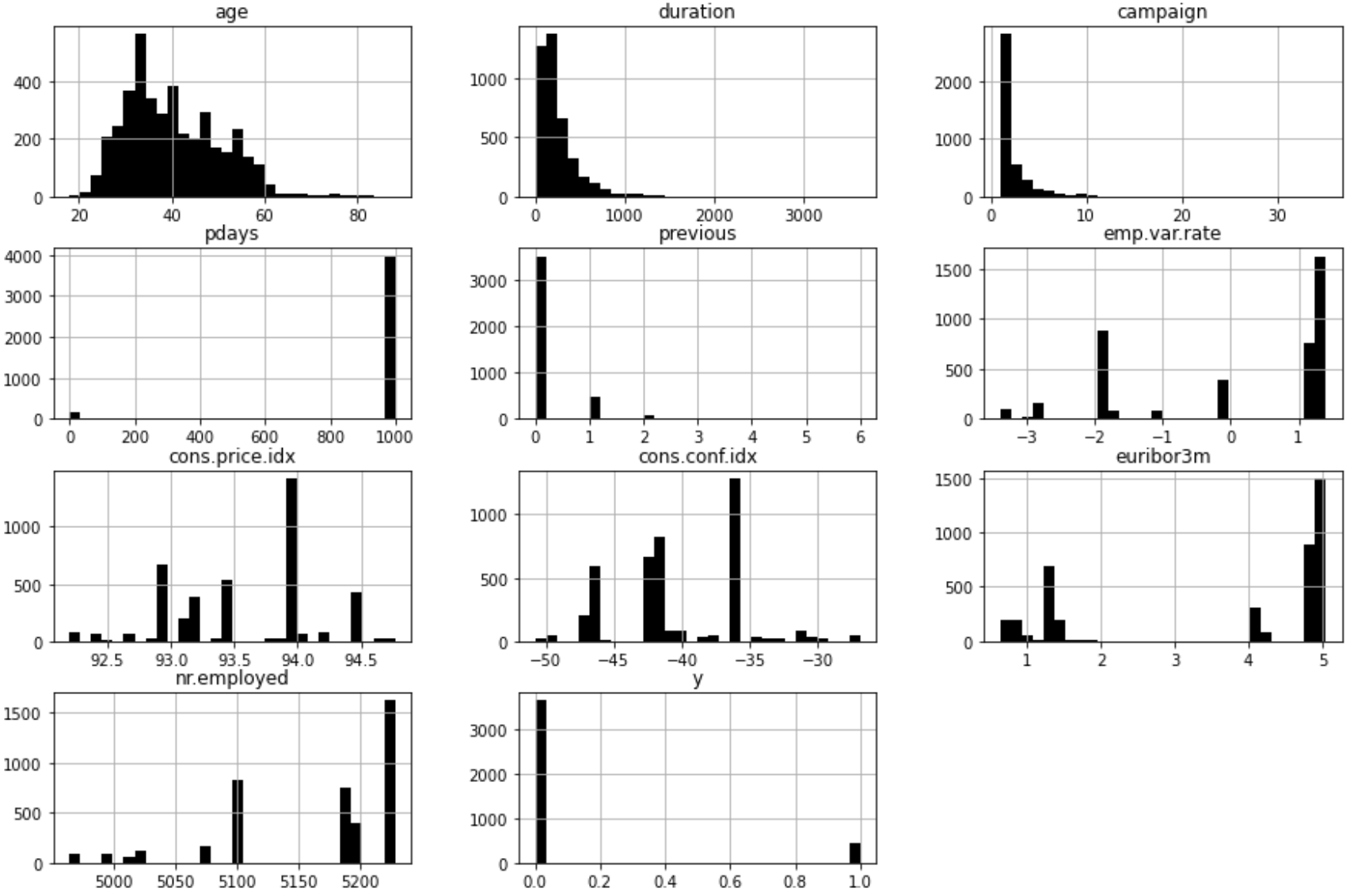
In [30]:
```
# creating a histogram for age
df["age"].hist()
```

Out[30]: <AxesSubplot:>

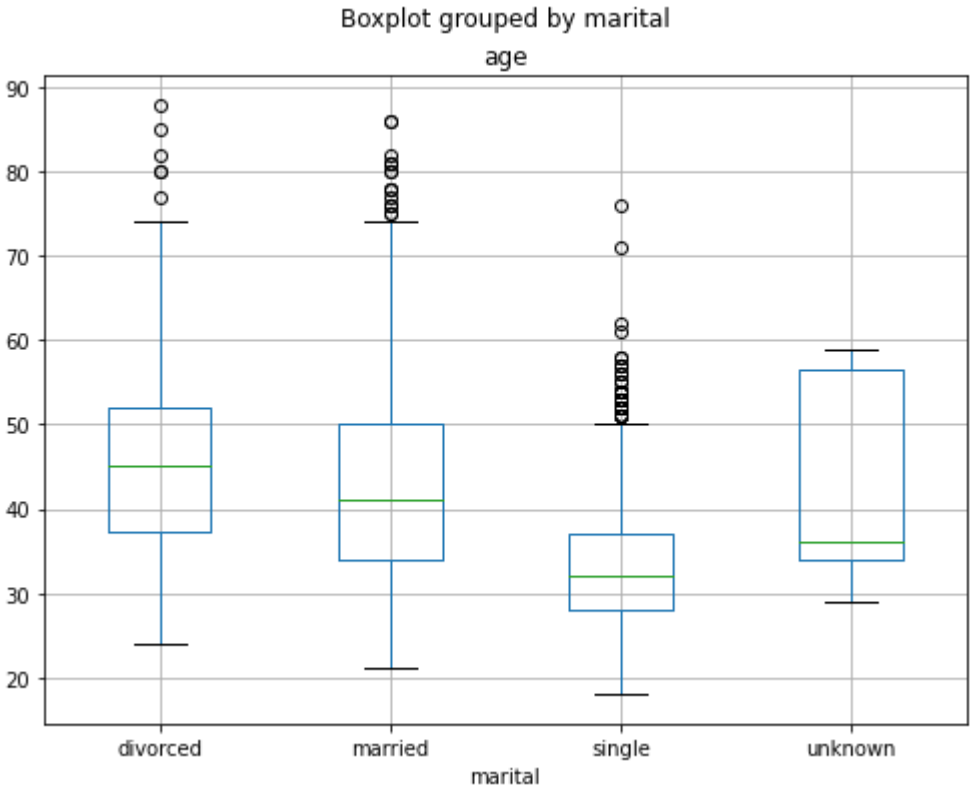


Now we will build histogram for features all together:

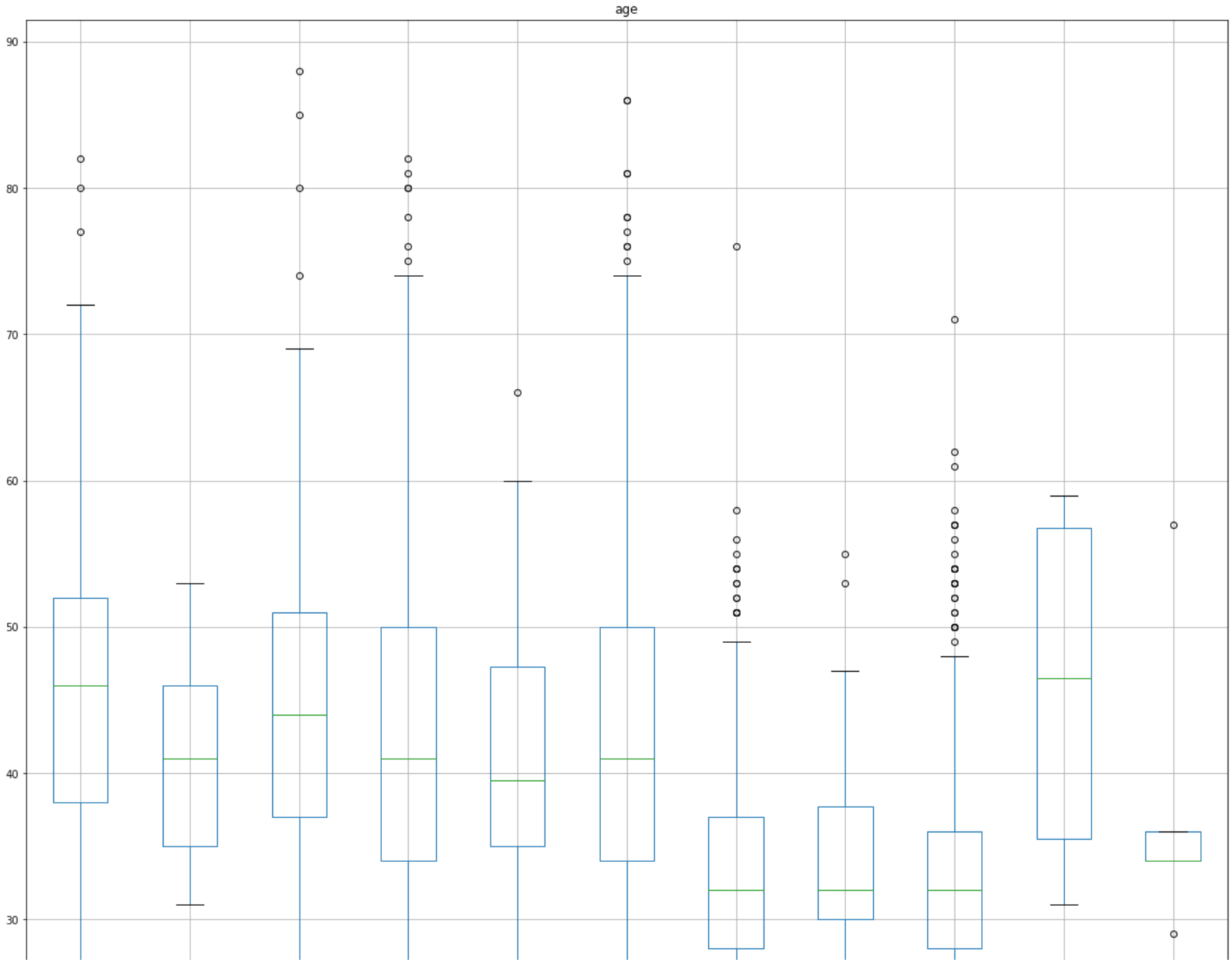
In [31]:
```
# collective histogram for all the features
df.hist(color = "k",
        bins = 30,
        figsize = (15, 10))
plt.show()
```

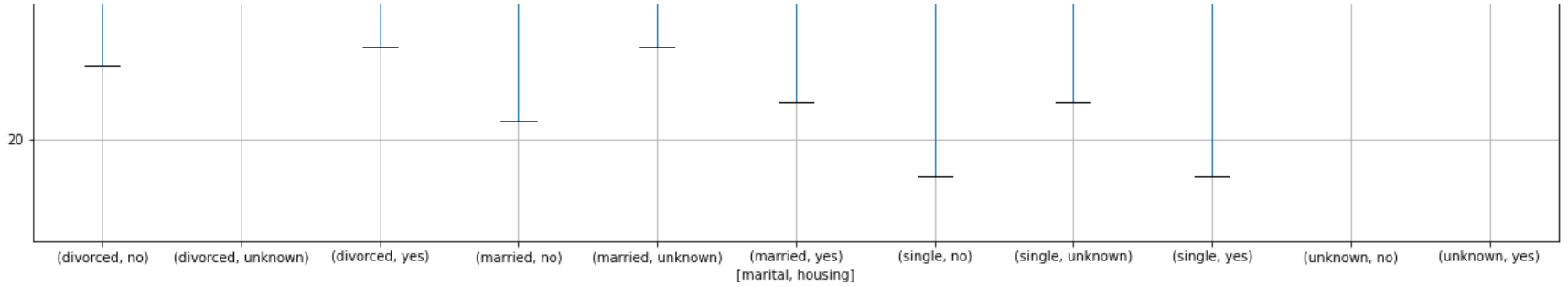


In [32]:
```
# creating a pandas boxplot
df.boxplot(column = "age",
           by = "marital")
plt.show()
```



In [33]:
```
df.boxplot(column = "age",
           by = ["marital", "housing"],
           figsize = (20, 20))
plt.show()
```


Boxplot grouped by ['marital', 'housing']
age
90
80
70
60
50
40
30



In [34]:
```
# top ten largest clietns of the bank
df.sort_values(by = "campaign", ascending = False).head(10)
```

Out[34]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	
2552	31	services	single	high.school	no	no	no	cellular	jul	thu	...	35	999	0	n
3564	25	admin.	single	basic.9y	no	no	no	cellular	jul	thu	...	29	999	0	n
3241	39	services	married	high.school	no	yes	no	cellular	jul	thu	...	29	999	0	n
56	29	admin.	single	university.degree	no	yes	no	telephone	jun	fri	...	27	999	0	n
2485	41	technician	married	high.school	no	yes	no	telephone	jun	fri	...	24	999	0	n
2988	45	services	married	professional.course	no	yes	no	cellular	jul	mon	...	23	999	0	n
2202	29	technician	married	university.degree	no	no	no	cellular	jul	thu	...	23	999	0	n
713	43	admin.	married	high.school	no	yes	no	cellular	jul	mon	...	22	999	0	n
3569	31	admin.	single	high.school	no	no	no	telephone	may	thu	...	22	999	0	n
886	56	technician	married	university.degree	unknown	no	no	cellular	jul	mon	...	19	999	0	n

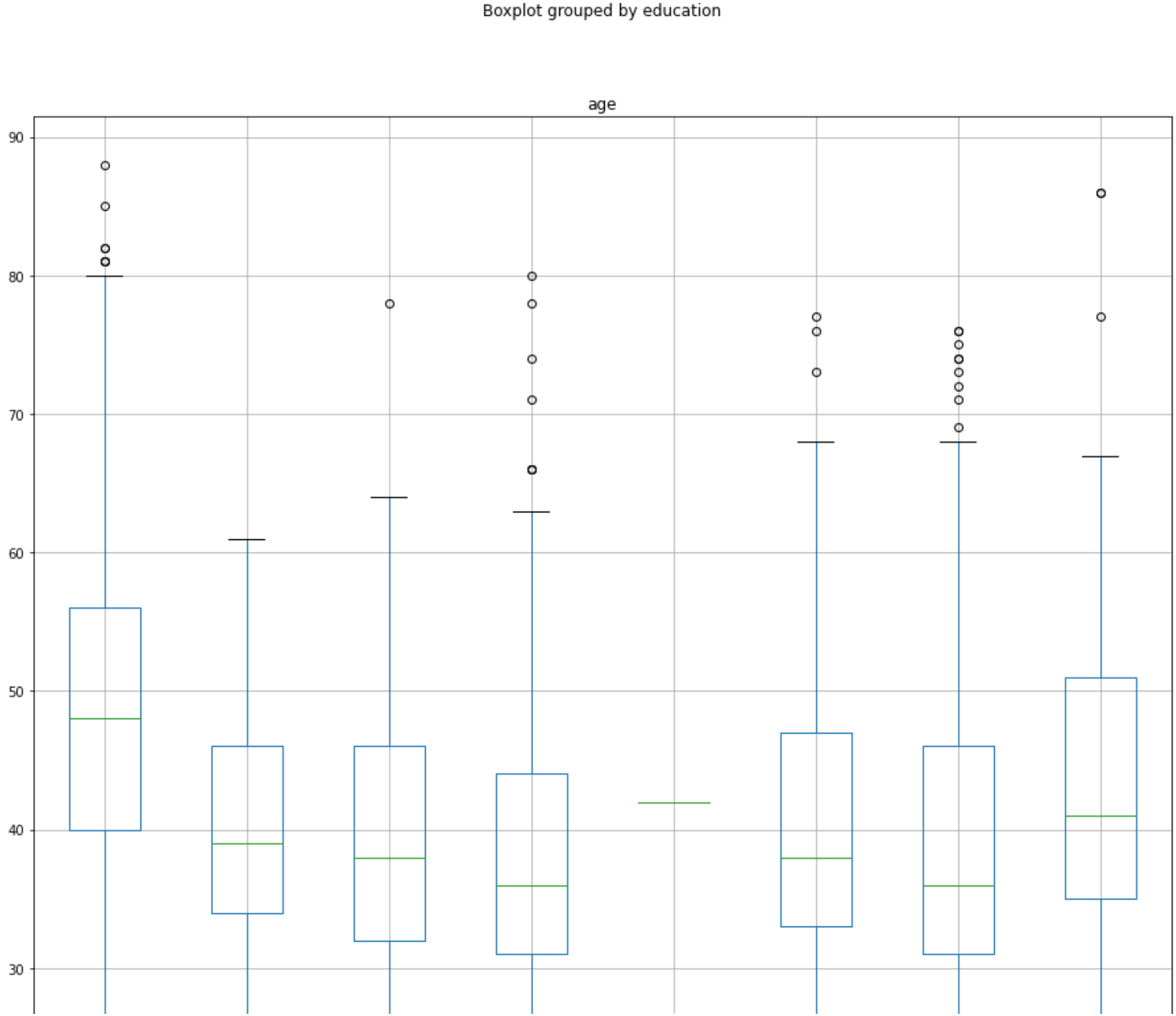
10 rows × 21 columns

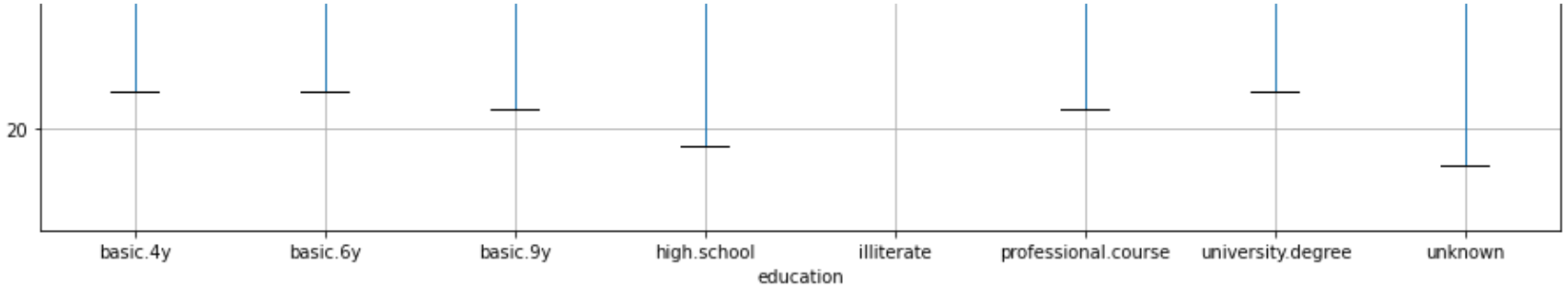
In [35]:
```
# client education median age and contacts
df.pivot_table(
    ["age", "campaign"],
    ["education"],
    aggfunc = ["mean", "count"],
)
```

Out[35]:

	mean		count	
	age	campaign	age	campaign
education				
basic.4y	47.657343	2.421911	429	429
basic.6y	40.144737	2.649123	228	228
basic.9y	39.231707	2.348432	574	574
high.school	38.097720	2.630836	921	921
illiterate	42.000000	4.000000	1	1
professional.course	40.207477	2.512150	535	535
university.degree	39.017405	2.583070	1264	1264
unknown	42.826347	2.538922	167	167

In [36]:
```
# boxplot to analyse age of client vs education
df.boxplot(column = "age",
  by = "education",
  figsize = (15, 15))
plt.show()
```


Boxplot grouped by education
age
90
80
70
60
50
40
30


20
basic.4y
basic.6y
basic.9y
high.school
illiterate
professional.course
university.degree
unknown
education