CreatingBinaryVariables

August 8, 2023

1 Creating Binary Variables

1.0.1 Import the Packages

[3]:
```
import pandas as pd
import numpy as np
import os
```

1.0.2 Load the Dataset

[4]:
```
filename = os.path.join(os.getcwd(), "data", "adult.data.partial")
df = pd.read_csv(filename, header=0)
```

1.0.3 Glance at the Data

[5]:
```
df.head()
```

[5]:

	age	workclass	fnlwgt	education	education-num	marital-status	\
0	36	State-gov	112074	Doctorate	16	Never-married	
1	35	Private	32528	HS-grad	9	Married-civ-spouse	
2	21	Private	270043	Some-college	10	Never-married	
3	45	Private	168837	Some-college	10	Married-civ-spouse	
4	39	Private	297449	Bachelors	13	Married-civ-spouse	

	occupation	relationship	race	sex_selfID	capital-gain	\
0	Prof-specialty	Not-in-family	White	Non-Female	0	
1	Handlers-cleaners	Husband	White	Non-Female	0	
2	Other-service	Own-child	White	Female	0	
3	Adm-clerical	Wife	White	Female	0	
4	Prof-specialty	Husband	White	Non-Female	0	

	capital-loss	hours-per-week	native-country	label
0	0	45	United-States	<=50K
1	0	45	United-States	<=50K
2	0	16	United-States	<=50K
3	0	24	Canada	>50K

```
4             0              40  United-States   >50K
```

1.0.4 Display Summary Statistics by Column

[6]:
```
df.describe(include='all')
```

[6]:
```
                age workclass        fnlwgt education  education-num  \
count   7000.000000      6625  7.000000e+03      7000    7000.000000
unique          NaN         7           NaN        16            NaN
top             NaN   Private           NaN   HS-grad            NaN
freq            NaN      4879           NaN      2263            NaN
mean      38.596714       NaN  1.924335e+05       NaN      10.049857
std       13.745594       NaN  1.063365e+05       NaN       2.580982
min       17.000000       NaN  1.882700e+04       NaN       1.000000
25%       28.000000       NaN  1.202478e+05       NaN       9.000000
50%       37.000000       NaN  1.821170e+05       NaN      10.000000
75%       47.000000       NaN  2.402370e+05       NaN      12.000000
max       90.000000       NaN  1.268339e+06       NaN      16.000000

           marital-status      occupation relationship   race  sex_selfID  \
count                7000            6625         7000   7000        7000
unique                  7              14            6      5           2
top     Married-civ-spouse  Prof-specialty      Husband  White  Non-Female
freq                 3277             911         2878   5990        4731
mean                  NaN             NaN          NaN    NaN         NaN
std                   NaN             NaN          NaN    NaN         NaN
min                   NaN             NaN          NaN    NaN         NaN
25%                   NaN             NaN          NaN    NaN         NaN
50%                   NaN             NaN          NaN    NaN         NaN
75%                   NaN             NaN          NaN    NaN         NaN
max                   NaN             NaN          NaN    NaN         NaN

        capital-gain  capital-loss  hours-per-week native-country  label
count    7000.000000   7000.000000     7000.000000           6862   7000
unique           NaN           NaN             NaN             40      2
top              NaN           NaN             NaN  United-States  <=50K
freq             NaN           NaN             NaN           6233   5319
mean     1079.000429     84.970286       40.107143            NaN    NaN
std      7011.160679    400.142351       12.323946            NaN    NaN
min         0.000000      0.000000        1.000000            NaN    NaN
25%         0.000000      0.000000       40.000000            NaN    NaN
50%         0.000000      0.000000       40.000000            NaN    NaN
75%         0.000000      0.000000       45.000000            NaN    NaN
max     99999.000000   4356.000000       99.000000            NaN    NaN
```

1.1 Group Columns into Binary Values Using `np.where()`

One of the typical tasks in data cleaning and feature engineering is to convert a feature or a label that has multiple categorical values into one that has a binary value. This means that instead of having multiple potential values, a feature or a label will have just two potential values.

For example, let's say we have "type of animal" as a feature and a list of 11 potential values for animals:

cats, dogs, sharks, elephants, iguanas, pigeons, humans, dolphins, mice, goldfish, hummingbirds

We can group these animals into **two** categories- `mammal` and `not-mammal` - and therefore, the "type of animal" feature will have two potential values (rather than 11 potential values).

cats, dogs, elephants, humans, dolphins, mice -> **mammal**

iguanas, pigeons, goldfish, sharks, hummingbirds -> **not-mammal**

In this exercise, we will convert one feature column in our dataset (the `workclass` column) to contain binary values.

In the cell below, inspect the current the `workclass` column in the DataFrame `df`. Notice the different feature values.

[7]:
```
df['workclass'].unique()
```

[7]:
```
array(['State-gov', 'Private', nan, 'Self-emp-not-inc', 'Local-gov',
       'Self-emp-inc', 'Federal-gov', 'Without-pay'], dtype=object)
```

Inspect the first 30 rows of DataFrame `df` and focus on the `workclass` column to see some of the above values.

[8]:
```
df.head(30)
```

[8]:

	age	workclass	fnlwgt	education	education-num	\
0	36	State-gov	112074	Doctorate	16	
1	35	Private	32528	HS-grad	9	
2	21	Private	270043	Some-college	10	
3	45	Private	168837	Some-college	10	
4	39	Private	297449	Bachelors	13	
5	27	Private	233421	Some-college	10	
6	40	Private	220460	HS-grad	9	
7	71	Private	163385	Some-college	10	
8	20	NaN	193416	Some-college	10	
9	41	Private	116391	Some-college	10	
10	54	Private	155408	HS-grad	9	
11	27	Self-emp-not-inc	140863	9th	5	
12	43	Private	170214	Some-college	10	
13	44	Private	198096	Bachelors	13	
14	32	Private	29933	Bachelors	13	
15	39	Self-emp-not-inc	142573	Some-college	10	
16	47	Local-gov	285060	Masters	14	
17	41	Private	56795	Masters	14	
18	34	State-gov	44464	Doctorate	16	
19	46	Private	141483	HS-grad	9	
20	49	Private	328776	HS-grad	9	
21	40	Self-emp-not-inc	165815	Some-college	10	

```
22   37           Private  167415       HS-grad             9
23   24           Private  388093     Bachelors            13
24   25           Private  166971       HS-grad             9
25   62           Private  588484     Bachelors            13
26   37           Private  155064     Assoc-voc            11
27   54  Self-emp-not-inc  205066          10th             6
28   28               NaN   55950     Bachelors            13
29   42      Self-emp-inc  152071   Prof-school            15

       marital-status         occupation   relationship   race  sex_selfID  \
0       Never-married     Prof-specialty  Not-in-family  White  Non-Female
1   Married-civ-spouse  Handlers-cleaners        Husband  White  Non-Female
2       Never-married      Other-service      Own-child  White      Female
3   Married-civ-spouse       Adm-clerical           Wife  White      Female
4   Married-civ-spouse     Prof-specialty        Husband  White  Non-Female
5       Never-married       Adm-clerical      Own-child  White  Non-Female
6       Never-married       Craft-repair  Not-in-family  White  Non-Female
7             Widowed              Sales  Not-in-family  White  Non-Female
8       Never-married                NaN      Own-child  White      Female
9   Married-civ-spouse       Craft-repair        Husband  White  Non-Female
10            Widowed  Handlers-cleaners      Unmarried  White      Female
11  Married-civ-spouse       Craft-repair        Husband  White  Non-Female
12  Married-civ-spouse    Exec-managerial        Husband  White  Non-Female
13  Married-civ-spouse              Sales        Husband  White  Non-Female
14  Married-civ-spouse  Handlers-cleaners        Husband  White  Non-Female
15      Never-married       Craft-repair  Not-in-family  White  Non-Female
16  Married-civ-spouse    Exec-managerial        Husband  White  Non-Female
17  Married-civ-spouse     Prof-specialty           Wife  White      Female
18      Never-married     Prof-specialty  Not-in-family  White      Female
19  Married-civ-spouse              Sales           Wife  White      Female
20           Divorced    Exec-managerial      Unmarried  White      Female
21      Never-married       Craft-repair      Own-child  White  Non-Female
22  Married-civ-spouse              Sales        Husband  White  Non-Female
23      Never-married    Exec-managerial  Not-in-family  Black  Non-Female
24  Married-civ-spouse              Sales        Husband  White  Non-Female
25  Married-civ-spouse              Sales        Husband  White  Non-Female
26           Divorced     Prof-specialty      Unmarried  White      Female
27  Married-civ-spouse       Craft-repair        Husband  White  Non-Female
28      Never-married                NaN      Own-child  Black      Female
29  Married-civ-spouse     Prof-specialty        Husband  White  Non-Female

    capital-gain  capital-loss  hours-per-week native-country  label
0              0             0              45  United-States  <=50K
1              0             0              45  United-States  <=50K
2              0             0              16  United-States  <=50K
3              0             0              24         Canada   >50K
4              0             0              40  United-States   >50K
```

5	0	0	20	United-States	<=50K
6	0	0	40	Canada	<=50K
7	0	0	35	United-States	>50K
8	0	0	40	United-States	<=50K
9	0	0	40	United-States	<=50K
10	0	0	40	United-States	<=50K
11	0	0	40	United-States	<=50K
12	0	0	50	United-States	>50K
13	7688	0	40	United-States	>50K
14	0	0	50	United-States	>50K
15	0	0	25	United-States	<=50K
16	0	1977	41	United-States	>50K
17	0	0	45	England	<=50K
18	0	0	60	United-States	<=50K
19	0	0	40	United-States	<=50K
20	0	0	40	United-States	<=50K
21	0	0	40	United-States	<=50K
22	0	0	50	United-States	<=50K
23	0	0	40	United-States	<=50K
24	0	0	52	United-States	<=50K
25	0	0	20	United-States	>50K
26	0	0	40	United-States	<=50K
27	0	0	36	United-States	<=50K
28	0	0	40	Germany	<=50K
29	0	0	60	Cuba	>50K

Our goal is to consolidate all of the different types of employment into two groups: **self-employed** and **not self-employed**, and change the values in the `workclass` column accordingly.

1.1.1 Step 1: Create Group 1: `Not-self-emp`

Notice that the `workclass` column contains the values `Self-emp-inc`and `Self-emp-not-inc` to indicate that an individual is self-employed (these correspond to incorporated and the unincorporated self-employment).

Note that the `workclass` column contains a number of different values for individuals who are *not* self employed ('State-gov', 'Private', 'Local-gov', 'Federal-gov', 'Without-pay').

As a first step, we can group all of the "not self employed" values into one category - a **not self employed** category. We can change all of the `workclass` columns that contain the values 'State-gov', 'Private', 'Local-gov', 'Federal-gov', 'Without-pay' to the value `'Not-self-emp'`.

Let's try this using the `np.where()` function. First, read the documentation for the `np.where()` function. Then examine and run the code below and try to understand how it works.

For more information about `np.where()`, consult the online documentation.

[9]:
```
np.where?
```

[10]:
```
# Since there are only two values for self-employment, we can simplify our code␣
↪by writing
# NOT self employed
```

```
# get all examples (rows) in which the workclass feature (column) is not␣
 ↪self-employed
# Note: the code below uses the Pandas logical operator ~ for NOT
columns_not_self_employed = ~(df['workclass'] == 'Self-emp-not-inc') &␣
 ↪~(df['workclass'] == 'Self-emp-inc')

#leave nan (null) in the dataset for now. Get all examples (rows) in which the␣
 ↪workclass feature (column) is not null
columns_not_null = ~(df['workclass'].isnull())

# create condition
condition = columns_not_self_employed & columns_not_null

# Use np.where() to change all of the workclass values that fulfill the␣
 ↪specified condition to Not-self-emp
df['workclass'] = np.where(condition, 'Not-self-emp', df['workclass'])

# Inspect the data to see the changed values
df.head(30)
```

[10]:

	age	workclass	fnlwgt	education	education-num	\
0	36	Not-self-emp	112074	Doctorate	16	
1	35	Not-self-emp	32528	HS-grad	9	
2	21	Not-self-emp	270043	Some-college	10	
3	45	Not-self-emp	168837	Some-college	10	
4	39	Not-self-emp	297449	Bachelors	13	
5	27	Not-self-emp	233421	Some-college	10	
6	40	Not-self-emp	220460	HS-grad	9	
7	71	Not-self-emp	163385	Some-college	10	
8	20	NaN	193416	Some-college	10	
9	41	Not-self-emp	116391	Some-college	10	
10	54	Not-self-emp	155408	HS-grad	9	
11	27	Self-emp-not-inc	140863	9th	5	
12	43	Not-self-emp	170214	Some-college	10	
13	44	Not-self-emp	198096	Bachelors	13	
14	32	Not-self-emp	29933	Bachelors	13	
15	39	Self-emp-not-inc	142573	Some-college	10	
16	47	Not-self-emp	285060	Masters	14	
17	41	Not-self-emp	56795	Masters	14	
18	34	Not-self-emp	44464	Doctorate	16	
19	46	Not-self-emp	141483	HS-grad	9	
20	49	Not-self-emp	328776	HS-grad	9	
21	40	Self-emp-not-inc	165815	Some-college	10	
22	37	Not-self-emp	167415	HS-grad	9	
23	24	Not-self-emp	388093	Bachelors	13	
24	25	Not-self-emp	166971	HS-grad	9	
25	62	Not-self-emp	588484	Bachelors	13	

```
26   37      Not-self-emp  155064     Assoc-voc             11
27   54  Self-emp-not-inc  205066          10th              6
28   28               NaN   55950     Bachelors             13
29   42      Self-emp-inc  152071   Prof-school             15

       marital-status         occupation   relationship   race  sex_selfID  \
0       Never-married     Prof-specialty  Not-in-family  White  Non-Female
1   Married-civ-spouse  Handlers-cleaners        Husband  White  Non-Female
2       Never-married      Other-service      Own-child  White      Female
3   Married-civ-spouse       Adm-clerical           Wife  White      Female
4   Married-civ-spouse     Prof-specialty        Husband  White  Non-Female
5       Never-married       Adm-clerical      Own-child  White  Non-Female
6       Never-married       Craft-repair  Not-in-family  White  Non-Female
7             Widowed              Sales  Not-in-family  White  Non-Female
8       Never-married                NaN      Own-child  White      Female
9   Married-civ-spouse       Craft-repair        Husband  White  Non-Female
10            Widowed  Handlers-cleaners      Unmarried  White      Female
11  Married-civ-spouse       Craft-repair        Husband  White  Non-Female
12  Married-civ-spouse    Exec-managerial        Husband  White  Non-Female
13  Married-civ-spouse              Sales        Husband  White  Non-Female
14  Married-civ-spouse  Handlers-cleaners        Husband  White  Non-Female
15      Never-married       Craft-repair  Not-in-family  White  Non-Female
16  Married-civ-spouse    Exec-managerial        Husband  White  Non-Female
17  Married-civ-spouse     Prof-specialty           Wife  White      Female
18      Never-married     Prof-specialty  Not-in-family  White      Female
19  Married-civ-spouse              Sales           Wife  White      Female
20           Divorced    Exec-managerial      Unmarried  White      Female
21      Never-married       Craft-repair      Own-child  White  Non-Female
22  Married-civ-spouse              Sales        Husband  White  Non-Female
23      Never-married    Exec-managerial  Not-in-family  Black  Non-Female
24  Married-civ-spouse              Sales        Husband  White  Non-Female
25  Married-civ-spouse              Sales        Husband  White  Non-Female
26           Divorced     Prof-specialty      Unmarried  White      Female
27  Married-civ-spouse       Craft-repair        Husband  White  Non-Female
28      Never-married                NaN      Own-child  Black      Female
29  Married-civ-spouse     Prof-specialty        Husband  White  Non-Female

    capital-gain  capital-loss  hours-per-week native-country  label
0              0             0              45  United-States  <=50K
1              0             0              45  United-States  <=50K
2              0             0              16  United-States  <=50K
3              0             0              24         Canada   >50K
4              0             0              40  United-States   >50K
5              0             0              20  United-States  <=50K
6              0             0              40         Canada  <=50K
7              0             0              35  United-States   >50K
8              0             0              40  United-States  <=50K
```

9	0	0	40	United-States	<=50K
10	0	0	40	United-States	<=50K
11	0	0	40	United-States	<=50K
12	0	0	50	United-States	>50K
13	7688	0	40	United-States	>50K
14	0	0	50	United-States	>50K
15	0	0	25	United-States	<=50K
16	0	1977	41	United-States	>50K
17	0	0	45	England	<=50K
18	0	0	60	United-States	<=50K
19	0	0	40	United-States	<=50K
20	0	0	40	United-States	<=50K
21	0	0	40	United-States	<=50K
22	0	0	50	United-States	<=50K
23	0	0	40	United-States	<=50K
24	0	0	52	United-States	<=50K
25	0	0	20	United-States	>50K
26	0	0	40	United-States	<=50K
27	0	0	36	United-States	<=50K
28	0	0	40	Germany	<=50K
29	0	0	60	Cuba	>50K

Run the cell below to see the new feature values for the `workclass` feature.

[11]:
```
df['workclass'].unique()
```

[11]:
```
array(['Not-self-emp', nan, 'Self-emp-not-inc', 'Self-emp-inc'],
      dtype=object)
```

Notice that we did not lose the `nan` values. Think about what would have happened had we not excluded 'NaN' values in our 'condition'.

1.1.2 Step 2: Create Group 2: `Self-emp`

We have successfully created a `Not-self-emp` group and changed all pertinent values in column `workclass` to the value `Not-self-emp`.

However, recall that our goal was to consolidate all of the different types of employment into two groups: **self-employed** and **not self-employed**

Notice that we still have two values for self employment: `Self-emp-not-inc`, `Self-emp-inc`.

We do not care to distinguish between the incorporated and the unincorporated self-employed examples, so we will next consolidate and change all `Self-emp-not-inc`and `Self-emp-inc` values in the `workclass` column to the value `Self-emp`.

In the two code cells below, you will use `np.where()` to combine `Self-emp-not-inc` and `Self-emp-inc` into a single class called `Self-emp`. Note: Follow the two steps as outlined per cell in order to properly grade your work.

Part 1: Create the condition and name it `condition`. `condition` will contain a compound statement joined together by the Boolean operator or (`|`). It will find the people who are self employed, i.e. it will look for both the value `Self-emp-not-inc` and the value `Self-emp-inc` in the `workclass` column.

1.1.3 Graded Cells

The two cells below will be graded. Remove the lines "raise NotImplementedError()" before writing your code.

[12]:
```
# YOUR CODE HERE
condition=(df['workclass']=='Self-emp-not-inc')|(df['workclass']=='Self-emp-inc')
```

1.1.4 Self-Check

Run the cell below to test the correctness of your code above before submitting for grading. Do not add code or delete code in the cell.

[13]:
```
# Run this self-test cell to check your code;
# do not add code or delete code in this cell
from jn import testCondition

try:
    p, err = testCondition(df, condition)
    print(err)
except Exception as e:
    print("Error!\n" + str(e))
```

```
Correct!
```

Part 2: In the code cell below, use `np.where()` to combine `Self-emp-not-inc` and `Self-emp-inc` into a single class called `Self-emp`. Use the condition you created in Step 1. Follow the code pattern you have seen previously when implementing `np.where()`.

[15]:
```
# YOUR CODE HERE
df['workclass']=np.where(condition,'Self-emp',df['workclass'])
```

1.1.5 Self-Check

Run the cell below to test the correctness of your code above before submitting for grading. Do not add code or delete code in the cell.

[16]:
```
# Run this self-test cell to check your code;
# do not add code or delete code in this cell
from jn import testWorkclass

try:
    p, err = testWorkclass(df)
    print(err)
except Exception as e:
    print("Error!\n" + str(e))
```

```
Correct!
```

[17]:
```
# Check your results
df['workclass'].unique()
```

[17]:
```
array(['Not-self-emp', nan, 'Self-emp'], dtype=object)
```

The cell below transforms the labels into a binary 'True' and 'False' variable, where 'True' is the label assigned to income level >50K. Run the code and inspect the results.

[18]:
```
condition1 = (df['label'] == '>50K')
df['label'] = np.where(condition1, 'True', df['label'])

condition2 = (df['label'] == '<=50K')
df['label'] = np.where(condition2, 'False', df['label'])

df.head(30)
```

[18]:

	age	workclass	fnlwgt	education	education-num	\
0	36	Not-self-emp	112074	Doctorate	16	
1	35	Not-self-emp	32528	HS-grad	9	
2	21	Not-self-emp	270043	Some-college	10	
3	45	Not-self-emp	168837	Some-college	10	
4	39	Not-self-emp	297449	Bachelors	13	
5	27	Not-self-emp	233421	Some-college	10	
6	40	Not-self-emp	220460	HS-grad	9	
7	71	Not-self-emp	163385	Some-college	10	
8	20	NaN	193416	Some-college	10	
9	41	Not-self-emp	116391	Some-college	10	
10	54	Not-self-emp	155408	HS-grad	9	
11	27	Self-emp	140863	9th	5	
12	43	Not-self-emp	170214	Some-college	10	
13	44	Not-self-emp	198096	Bachelors	13	
14	32	Not-self-emp	29933	Bachelors	13	
15	39	Self-emp	142573	Some-college	10	
16	47	Not-self-emp	285060	Masters	14	
17	41	Not-self-emp	56795	Masters	14	
18	34	Not-self-emp	44464	Doctorate	16	
19	46	Not-self-emp	141483	HS-grad	9	
20	49	Not-self-emp	328776	HS-grad	9	
21	40	Self-emp	165815	Some-college	10	
22	37	Not-self-emp	167415	HS-grad	9	
23	24	Not-self-emp	388093	Bachelors	13	
24	25	Not-self-emp	166971	HS-grad	9	
25	62	Not-self-emp	588484	Bachelors	13	
26	37	Not-self-emp	155064	Assoc-voc	11	
27	54	Self-emp	205066	10th	6	
28	28	NaN	55950	Bachelors	13	
29	42	Self-emp	152071	Prof-school	15	

	marital-status	occupation	relationship	race	sex_selfID	\

0	Never-married	Prof-specialty	Not-in-family	White	Non-Female	
1	Married-civ-spouse	Handlers-cleaners	Husband	White	Non-Female	
2	Never-married	Other-service	Own-child	White	Female	
3	Married-civ-spouse	Adm-clerical	Wife	White	Female	
4	Married-civ-spouse	Prof-specialty	Husband	White	Non-Female	
5	Never-married	Adm-clerical	Own-child	White	Non-Female	
6	Never-married	Craft-repair	Not-in-family	White	Non-Female	
7	Widowed	Sales	Not-in-family	White	Non-Female	
8	Never-married	NaN	Own-child	White	Female	
9	Married-civ-spouse	Craft-repair	Husband	White	Non-Female	
10	Widowed	Handlers-cleaners	Unmarried	White	Female	
11	Married-civ-spouse	Craft-repair	Husband	White	Non-Female	
12	Married-civ-spouse	Exec-managerial	Husband	White	Non-Female	
13	Married-civ-spouse	Sales	Husband	White	Non-Female	
14	Married-civ-spouse	Handlers-cleaners	Husband	White	Non-Female	
15	Never-married	Craft-repair	Not-in-family	White	Non-Female	
16	Married-civ-spouse	Exec-managerial	Husband	White	Non-Female	
17	Married-civ-spouse	Prof-specialty	Wife	White	Female	
18	Never-married	Prof-specialty	Not-in-family	White	Female	
19	Married-civ-spouse	Sales	Wife	White	Female	
20	Divorced	Exec-managerial	Unmarried	White	Female	
21	Never-married	Craft-repair	Own-child	White	Non-Female	
22	Married-civ-spouse	Sales	Husband	White	Non-Female	
23	Never-married	Exec-managerial	Not-in-family	Black	Non-Female	
24	Married-civ-spouse	Sales	Husband	White	Non-Female	
25	Married-civ-spouse	Sales	Husband	White	Non-Female	
26	Divorced	Prof-specialty	Unmarried	White	Female	
27	Married-civ-spouse	Craft-repair	Husband	White	Non-Female	
28	Never-married	NaN	Own-child	Black	Female	
29	Married-civ-spouse	Prof-specialty	Husband	White	Non-Female	

	capital-gain	capital-loss	hours-per-week	native-country	label
0	0	0	45	United-States	False
1	0	0	45	United-States	False
2	0	0	16	United-States	False
3	0	0	24	Canada	True
4	0	0	40	United-States	True
5	0	0	20	United-States	False
6	0	0	40	Canada	False
7	0	0	35	United-States	True
8	0	0	40	United-States	False
9	0	0	40	United-States	False
10	0	0	40	United-States	False
11	0	0	40	United-States	False
12	0	0	50	United-States	True
13	7688	0	40	United-States	True
14	0	0	50	United-States	True

15	0	0	25	United-States	False
16	0	1977	41	United-States	True
17	0	0	45	England	False
18	0	0	60	United-States	False
19	0	0	40	United-States	False
20	0	0	40	United-States	False
21	0	0	40	United-States	False
22	0	0	50	United-States	False
23	0	0	40	United-States	False
24	0	0	52	United-States	False
25	0	0	20	United-States	True
26	0	0	40	United-States	False
27	0	0	36	United-States	False
28	0	0	40	Germany	False
29	0	0	60	Cuba	True

1.2 Deep Dive: Categorical Variables

Let us discuss categorical variables in a little bit more detail.

A categorical data variable is often more than just a variable who's value is one of two or more classes, or categories. Sometimes the *order* of categories is meaningful, and contains some information that may be useful for analysis. For example, we may have a dataset that contains a variable 'ice_cream_flavor' that takes five possible values: 'vanilla', 'strawberry', 'pistachio', 'chocolate', and 'mango'. In this example, there is not a natural order to the five categories. Now consider a variable called 'portion_size' that takes on one of the three possible values: 'standard', 'double', and 'super'. In this case, the three categories are related to one another by an *ordering*. We would like for any future model we fit to this data to recognize and make use of the fact that

$$standard < double < super.$$

Try to think of a variable in our dataset that should be an ordered categorical variable.

One such variable would be 'education'. Let's examine the list of possible values:

[19]:
```
df['education'].unique()
```

[19]:
```
array(['Doctorate', 'HS-grad', 'Some-college', 'Bachelors', '9th',
       'Masters', 'Assoc-voc', '10th', 'Prof-school', '7th-8th',
       'Assoc-acdm', '11th', '5th-6th', '1st-4th', '12th', 'Preschool'],
      dtype=object)
```

There is clearly an underlying order to these categories! In general, you would have to consult the data manual to establish what the correct order is. For this data, the true ordering would be:

$$Preschool < 1st-4th < 5th-6th < 7th-8th < 9th < 10th < 11th < 12th < HS-grad < Prof-school < Assoc$$

Is this variable currently ordered?

[20]:
```
df['education'].dtype
```

[20]:
```
dtype('O')
```

No! This variable is of type 'object'. Any future analysis in python won't recognize the underlying order of categories. We need to transform this variable into an ordered categorical variable. Here's how.

[21]:
```
# First, create a correctly ordered list of category names:
edu = ['Preschool', '1st-4th', '5th-6th', '7th-8th', '9th', '10th', '1th',␣
 ↪'12th', 'HS-grad', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Some-college',␣
 ↪'Bachelors', 'Masters', 'Doctorate']
# Then, use the pd.Categorical method to reassign the values of this column as␣
 ↪a new type with awareness of the order:
df['education'] = pd.Categorical(df['education'], ordered=True, categories=edu)
```

[22]:
```
df['education'].dtype
```

[22]:
```
CategoricalDtype(categories=['Preschool', '1st-4th', '5th-6th', '7th-8th',
'9th', '10th',
                  '1th', '12th', 'HS-grad', 'Prof-school', 'Assoc-acdm',
                  'Assoc-voc', 'Some-college', 'Bachelors', 'Masters',
                  'Doctorate'],
                 ordered=True)
```

Perfect! Now we can be sure that the order is recognized by Python. (Note: This will come in handy when we create plots with categories on one axis.)

1.2.1 Convert Categorical Variables into "Dummy" Variables

There are many different ways we can transform our categorical data into numerical data to prepare for the model training phase. Pandas has one function that helps us transform categorical values into binary ones. It is the `pd.get_dummies()` function. Often we refer to these binary values as "dummy" values, or variables. Read up on the Pandas `pd.get_dummies()` function. You can consult the online documentation. Run the cell below to observe the results of creating dummies for all variables in our data.

[23]:
```
df_binary = pd.get_dummies(df)
df_binary.head()
```

[23]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	\
0	36	112074	16	0	0	45	
1	35	32528	9	0	0	45	
2	21	270043	10	0	0	16	
3	45	168837	10	0	0	24	
4	39	297449	13	0	0	40	

	workclass_Not-self-emp	workclass_Self-emp	education_Preschool	\
0	1	0	0	
1	1	0	0	
2	1	0	0	
3	1	0	0	
4	1	0	0	

	education_1st-4th	...	native-country_Scotland	native-country_South	\

```
0                   0  ...                          0                    0
1                   0  ...                          0                    0
2                   0  ...                          0                    0
3                   0  ...                          0                    0
4                   0  ...                          0                    0

   native-country_Taiwan  native-country_Thailand  \
0                      0                        0
1                      0                        0
2                      0                        0
3                      0                        0
4                      0                        0

   native-country_Trinadad&Tobago  native-country_United-States  \
0                               0                             1
1                               0                             1
2                               0                             1
3                               0                             0
4                               0                             1

   native-country_Vietnam  native-country_Yugoslavia  label_False  label_True
0                       0                          0            1           0
1                       0                          0            1           0
2                       0                          0            1           0
3                       0                          0            0           1
4                       0                          0            0           1

[5 rows x 100 columns]
```

[ ]: