MissingData

August 8, 2023

1 Finding and Replacing Missing Data

[1]:
```
import pandas as pd
import numpy as np
import os
```

1.1 Load and Inspect the Data

[2]:
```
filename = os.path.join(os.getcwd(), "data", "adult.data.partial.missing")
df = pd.read_csv(filename, header=0)
```

[3]:
```
df.shape
```

[3]:
```
(7000, 15)
```

[4]:
```
df.head()
```

[4]:
```
    age  workclass  fnlwgt     education  education-num      marital-status  \
0  36.0  State-gov  112074     Doctorate             16       Never-married
1  35.0    Private   32528       HS-grad              9  Married-civ-spouse
2  21.0    Private  270043  Some-college             10       Never-married
3  45.0    Private  168837  Some-college             10  Married-civ-spouse
4  39.0    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            45.0  United-States  <=50K
1             0            45.0  United-States  <=50K
2             0            16.0  United-States  <=50K
3             0            24.0         Canada   >50K
4             0            40.0  United-States   >50K
```

1.2 Dealing with Missing Data

Our goal will be to identify which columns in a dataset have missing values, and to replace a missing value in a column with the mean of the other values in that column. We will add dummy variables to our dataset to indicate which columns initially had missing values.

1.2.1 Step 1: Identify Missing Values Using Pandas `isnull()` Method

First let us check if there are missing values in DataFrame `df`.

[5]:
```
df.isnull().values.any()
```

[5]:
```
True
```

DataFrame `df` contains missing values! The Pandas `isnull()` method returns `True`/`False` values indicating whether a value is or is not missing in a particular position in a DataFrame or Series. This method recognizes various spellings of missingness like `NaN`, `nan`, `None`, and `NA` among others. Consult the online documentation for more information.

[6]:
```
df.isnull().head()
```

[6]:
```
     age  workclass  fnlwgt  education  education-num  marital-status  \
0  False      False   False      False          False           False
1  False      False   False      False          False           False
2  False      False   False      False          False           False
3  False      False   False      False          False           False
4  False      False   False      False          False           False

   occupation  relationship   race  sex_selfID  capital-gain  capital-loss  \
0       False         False  False       False         False         False
1       False         False  False       False         False         False
2       False         False  False       False         False         False
3       False         False  False       False         False         False
4       False         False  False       False         False         False

   hours-per-week  native-country  label
0           False           False  False
1           False           False  False
2           False           False  False
3           False           False  False
4           False           False  False
```

The code cell below counts the number of times a missing value occurs in each column. It applies the `isnull()` method and then aggregates the results by columns using the `np.sum()` function. For more information about `np.sum()`, consult the online documentation.

[7]:
```
nan_count = np.sum(df.isnull(), axis = 0)
nan_count
```

[7]:
```
age                35
workclass         375
fnlwgt              0
education           0
```

```
education-num       0
marital-status      0
occupation        375
relationship        0
race                0
sex_selfID          0
capital-gain        0
capital-loss        0
hours-per-week     70
native-country    138
label               0
dtype: int64
```

The code cell below stores the names of the columns with detected missing values into a Python list.

[8]:
```python
condition = nan_count != 0 # look for all columns with missing values

col_names = nan_count[condition].index # get the column names
print(col_names)

nan_cols = list(col_names) # convert column names to list
print(nan_cols)
```

```
Index(['age', 'workclass', 'occupation', 'hours-per-week', 'native-country'],
dtype='object')
['age', 'workclass', 'occupation', 'hours-per-week', 'native-country']
```

1.2.2 Step 2: Choose Which Values to Fill

We can see that five columns in our DataFrame contain missing values. Would you want to replace the missing values with something for every one of these columns?

Let's take a look at the data types of the columns that contain missing values using `dtypes`.

[9]:
```python
nan_col_types = df[nan_cols].dtypes
nan_col_types
```

[9]:
```
age                float64
workclass           object
occupation          object
hours-per-week     float64
native-country      object
dtype: object
```

For three of the five identified columns, the type is 'object'. Is this a problem? A common approach to dealing with the missing values is to replace those values with either the mean, the median, or some other type of 'representative' value wherever a `nan` occurs. This, of course, assumes that the column is numerical to begin with. That does not seem to be true for the `workclass`, `occupation`, and `native-country` variables. Let us confirm:

[10]:
```
print(df['workclass'].unique())
print(df['occupation'].unique())
print(df['native-country'].unique())
```

```
['State-gov' 'Private' nan 'Self-emp-not-inc' 'Local-gov' 'Self-emp-inc'
 'Federal-gov' 'Without-pay']
['Prof-specialty' 'Handlers-cleaners' 'Other-service' 'Adm-clerical'
 'Craft-repair' 'Sales' nan 'Exec-managerial' 'Farming-fishing'
 'Machine-op-inspct' 'Transport-moving' 'Tech-support' 'Priv-house-serv'
 'Protective-serv' 'Armed-Forces']
['United-States' 'Canada' 'England' 'Germany' 'Cuba' nan 'Puerto-Rico'
 'Mexico' 'Nicaragua' 'China' 'South' 'India' 'Vietnam' 'Philippines'
 'El-Salvador' 'Guatemala' 'Japan' 'Jamaica' 'Peru' 'France' 'Greece'
 'Italy' 'Columbia' 'Honduras' 'Iran' 'Poland' 'Haiti'
 'Dominican-Republic' 'Scotland' 'Yugoslavia' 'Trinadad&Tobago' 'Ireland'
 'Portugal' 'Taiwan' 'Hong' 'Ecuador' 'Laos' 'Hungary' 'Thailand'
 'Outlying-US(Guam-USVI-etc)' 'Cambodia']
```

The concept of 'mean' is not defined for string entries, so filling in the missing values with the mean of the column wouldn't work here. In real business settings, one way to go about filling in the missing values would be to fit a model that predicts the country based on other values. All data-filling methods come with caveats, and some may threaten the validity of your larger analytical conclusions.

For the rest of this exercise, we will focus only on the numerical variables, for which it makes sense to replace every missing value with the mean of the column. Those are `age` and `hours-per-week` columns.

1.2.3 Step 3: Create 'Dummy' Variables for Missing Values

No method of imputing missing values is perfect, and for this reason it makes sense to keep track of which values we artificially created.

The code cell below looks at the the values in columns `age` and `hours-per-week` and stores the corresponding `True/False` values (True if the value is missing and False if the value is present) in new columns `age_na` and `hours-per-week_na`. Run the cell and inspect the new columns.

[11]:
```
df['age_na'] = df['age'].isnull()
df['hours-per-week_na'] = df['hours-per-week'].isnull()
df.head()
```

[11]:
```
    age  workclass  fnlwgt     education  education-num      marital-status  \
0  36.0  State-gov  112074     Doctorate             16       Never-married
1  35.0    Private   32528       HS-grad              9  Married-civ-spouse
2  21.0    Private  270043  Some-college             10       Never-married
3  45.0    Private  168837  Some-college             10  Married-civ-spouse
4  39.0    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  age_na  \
0             0            45.0  United-States  <=50K   False
1             0            45.0  United-States  <=50K   False
2             0            16.0  United-States  <=50K   False
3             0            24.0         Canada   >50K   False
4             0            40.0  United-States   >50K   False

   hours-per-week_na
0              False
1              False
2              False
3              False
4              False
```

1.2.4 Step 4: Fill the Missing Values Using Pandas `fillna()` Method

The Pandas `fillna()` method is used to "fill in" missing values in a Series or DataFrame object. Consult the online documentation for more information about how to use the `fillna()` method. The code cell below uses `fillna()` to fill in values for the missing values in the `age` column. It fills in the missing values with the mean value of all of the existing values in the that column. It uses the Pandas `mean()` method to compute the replacement values. For more information about `mean()`, consult the online documentation.

Tip: when working with `fillna()`, make sure that you do not just create a copy object with the filled values, but change the original values of the `df` object by specifying the `inplace = True` parameter value.

First inspect some of the columns that contain missing values.

[12]:
```python
df.loc[df['age'].isnull()]
```

[12]:
```
       age         workclass  fnlwgt     education  education-num  \
453    NaN           Private  117166     Bachelors             13
654    NaN               NaN   65545       Masters             14
865    NaN  Self-emp-not-inc   93806  Some-college             10
1206   NaN           Private  441637       HS-grad              9
1262   NaN           Private  350440       HS-grad              9
1302   NaN           Private  317443  Some-college             10
1496   NaN           Private   99185       HS-grad              9
2100   NaN           Private  179271  Some-college             10
2581   NaN           Private  145160  Some-college             10
2651   NaN           Private  151580  Some-college             10
2961   NaN           Private  363219  Some-college             10
3174   NaN  Self-emp-not-inc   96245       HS-grad              9
3370   NaN           Private  214502           9th              5
3594   NaN           Private  265807  Some-college             10
```

```
3721  NaN           Private  322391          11th              7
3769  NaN         Local-gov   82393       HS-grad              9
3993  NaN           Private  232024          11th              7
3997  NaN           Private  235894          11th              7
4048  NaN           Private  202498          11th              7
4100  NaN           Private   33644       HS-grad              9
4253  NaN           Private  148524       HS-grad              9
4670  NaN  Self-emp-not-inc   29054       HS-grad              9
4802  NaN           Private  173208       Masters             14
4828  NaN           Private  191982     Assoc-voc             11
4866  NaN         Local-gov  286342       Masters             14
5299  NaN           Private  329426       Masters             14
5420  NaN       Federal-gov  239074    Assoc-acdm             12
5871  NaN           Private  298635       Masters             14
5949  NaN           Private  157894  Some-college             10
6007  NaN           Private  266635       HS-grad              9
6153  NaN           Private  236818     Assoc-voc             11
6219  NaN           Private   57916       HS-grad              9
6466  NaN           Private  223515       HS-grad              9
6596  NaN           Private  152307       HS-grad              9
6833  NaN           Private  289458     Bachelors             13

          marital-status         occupation   relationship  \
453        Never-married    Exec-managerial  Not-in-family
654             Divorced                NaN      Own-child
865   Married-civ-spouse              Sales        Husband
1206  Married-civ-spouse       Tech-support        Husband
1262  Married-civ-spouse    Exec-managerial        Husband
1302       Never-married       Adm-clerical      Own-child
1496  Married-civ-spouse              Sales        Husband
2100  Married-civ-spouse       Craft-repair        Husband
2581  Married-civ-spouse  Machine-op-inspct        Husband
2651  Married-civ-spouse     Prof-specialty        Husband
2961       Never-married      Other-service  Not-in-family
3174  Married-civ-spouse  Machine-op-inspct        Husband
3370  Married-civ-spouse  Handlers-cleaners        Husband
3594           Separated       Craft-repair  Not-in-family
3721           Separated      Other-service      Unmarried
3769       Never-married  Handlers-cleaners      Unmarried
3993       Never-married  Machine-op-inspct      Own-child
3997  Married-civ-spouse    Exec-managerial        Husband
4048  Married-civ-spouse  Handlers-cleaners        Husband
4100       Never-married       Adm-clerical      Own-child
4253  Married-civ-spouse   Transport-moving        Husband
4670  Married-civ-spouse    Farming-fishing        Husband
4802  Married-civ-spouse     Prof-specialty        Husband
4828       Never-married       Adm-clerical      Own-child
```

4866	Never-married	Prof-specialty	Not-in-family
5299	Never-married	Exec-managerial	Not-in-family
5420	Married-civ-spouse	Other-service	Husband
5871	Married-civ-spouse	Prof-specialty	Husband
5949	Never-married	Other-service	Own-child
6007	Never-married	Other-service	Own-child
6153	Never-married	Prof-specialty	Unmarried
6219	Separated	Farming-fishing	Own-child
6466	Married-civ-spouse	Craft-repair	Husband
6596	Married-civ-spouse	Machine-op-inspct	Husband
6833	Never-married	Exec-managerial	Not-in-family

	race	sex_selfID	capital-gain	capital-loss \
453	White	Non-Female	0	0
654	White	Female	0	0
865	White	Non-Female	0	0
1206	White	Non-Female	0	0
1262	Asian-Pac-Islander	Non-Female	0	0
1302	Black	Female	0	0
1496	White	Non-Female	7298	0
2100	White	Non-Female	0	0
2581	White	Non-Female	0	0
2651	White	Non-Female	15024	0
2961	White	Female	0	0
3174	White	Non-Female	0	0
3370	White	Non-Female	0	0
3594	White	Non-Female	0	0
3721	Black	Female	0	0
3769	Asian-Pac-Islander	Non-Female	0	1590
3993	White	Non-Female	0	0
3997	White	Non-Female	0	0
4048	White	Non-Female	0	0
4100	White	Female	0	0
4253	White	Non-Female	0	2057
4670	White	Non-Female	0	0
4802	White	Non-Female	0	0
4828	White	Female	0	0
4866	White	Female	0	0
5299	White	Non-Female	0	0
5420	White	Non-Female	0	0
5871	Asian-Pac-Islander	Non-Female	0	0
5949	Black	Non-Female	0	0
6007	Black	Non-Female	0	0
6153	Black	Female	0	0
6219	White	Non-Female	0	0
6466	White	Non-Female	0	0
6596	White	Non-Female	0	0

```
6833             White      Female            0             0

      hours-per-week native-country  label  age_na  hours-per-week_na
453             50.0  United-States  <=50K    True              False
654             55.0  United-States  <=50K    True              False
865             55.0  United-States  <=50K    True              False
1206            40.0  United-States  <=50K    True              False
1262            40.0  United-States   >50K    True              False
1302            15.0  United-States  <=50K    True              False
1496            50.0  United-States   >50K    True              False
2100            50.0  United-States   >50K    True              False
2581            43.0  United-States  <=50K    True              False
2651            40.0  United-States   >50K    True              False
2961            20.0  United-States  <=50K    True              False
3174            40.0  United-States  <=50K    True              False
3370            50.0  United-States   >50K    True              False
3594            45.0  United-States  <=50K    True              False
3721             NaN  United-States  <=50K    True               True
3769            45.0  United-States  <=50K    True              False
3993            55.0  United-States  <=50K    True              False
3997            55.0  United-States  <=50K    True              False
4048            40.0       Columbia  <=50K    True              False
4100            30.0  United-States  <=50K    True              False
4253            40.0  United-States  <=50K    True              False
4670            50.0  United-States  <=50K    True              False
4802            25.0  United-States  <=50K    True              False
4828            55.0  United-States  <=50K    True              False
4866            32.0  United-States   >50K    True              False
5299            37.0  United-States  <=50K    True              False
5420            40.0  United-States  <=50K    True              False
5871            40.0           Hong   >50K    True              False
5949            20.0  United-States  <=50K    True              False
6007            30.0  United-States  <=50K    True              False
6153            26.0  United-States  <=50K    True              False
6219            40.0  United-States  <=50K    True              False
6466            45.0  United-States  <=50K    True              False
6596            40.0  United-States  <=50K    True              False
6833            40.0  United-States  <=50K    True              False
```

[13]:

```
# look at one row that contains a missing value for age
print("Row 654:  " + str(df['age'][654]))

# compute mean for all non null age values
mean_ages=df['age'].mean()
print("mean value for all age columns: " + str(mean_ages))

# fill all missing values with the mean
```

```
df['age'].fillna(value=mean_ages, inplace=True)

# look at one of the rows that contained a missing value for age.
# It should now contain the mean
print("Row 654:  " + str(df['age'][654]))
```

```
Row 654:  nan
mean value for all age columns: 38.61981335247667
Row 654:  38.61981335247667
```

In the code cell below, do the same for the `hours-per-week` column.

1. Compute the mean value of the `hours-per-week` column and save the result to variable `mean_hours`
2. Use `fillna` to change the values of the missing columns to `mean_hours`.

1.2.5 Graded Cell

The cell below will be graded. Remove the line "raise NotImplementedError()" before writing your code.

[14]:
```
# YOUR CODE HERE
mean_hours=df['hours-per-week'].mean()
df['hours-per-week'].fillna(mean_hours,inplace=True)
```

1.2.6 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.

[15]:
```
# Run this self-test cell to check your code;
# do not add code or delete code in this cell
from jn import testFillNa

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

```
Correct!
```

Check if we successfully converted all missing values to the mean value. Display the sum of missing values for the age column.

[16]:
```
np.sum(df['age'].isnull(), axis = 0)
```

[16]:
```
0
```

In the code cell below, do the same for the `hours-per-week` column. Save the result to variable `sum_hours`.

1.2.7 Graded Cell

The cell below will be graded. Remove the line "raise NotImplementedError()" before writing your code.

[23]:
```
# YOUR CODE HERE
sum_hours=np.sum(df['hours-per-week'].isnull(),axis=0)
```

1.2.8 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.

[24]:
```
# Run this self-test cell to check your code;
# do not add code or delete code in this cell
from jn import testSumHours

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

```
Correct!
```

[ ]:
```

```