Data Wrangling

December 6, 2021

[1]:
```
#Data Wrangling
# It is the process of transforming raw data to a clean and organised format␣
↪ready to use

import pandas as pd

url = "https://osf.io/aupb4/download"

dataframe = pd.read_csv(url)
dataframe.head(5)
```

[1]:
```
                                           Name PClass    Age     Sex  \
0                  Allen, Miss Elisabeth Walton    1st  29.00  female
1                   Allison, Miss Helen Loraine    1st   2.00  female
2           Allison, Mr Hudson Joshua Creighton    1st  30.00    male
3 Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st  25.00  female
4                 Allison, Master Hudson Trevor    1st   0.92    male

   Survived
0         1
1         0
2         0
3         0
4         1
```

[2]:
```
#Creating Data Frame

dataframe = pd.DataFrame()

#Adding columns
dataframe["Name"]=["A","B","C"]
dataframe["Age"]=[23,12,8]
dataframe["Gender"]=["M","F","F"]
```

[3]:
```
#Create row
new_person = pd.Series(["M",22,"M"],index=["Name","Age","Gender"])
```

```
#Append row to datframe
dataframe = dataframe.append(new_person,ignore_index=True)
print(dataframe)
```

```
  Name  Age Gender
0    A   23      M
1    B   12      F
2    C    8      F
3    M   22      M
```

[4]:
```
#Describing the Data
url = "https://osf.io/aupb4/download"

dataframe = pd.read_csv(url)
print(dataframe.head(4))
```

```
                                            Name PClass   Age     Sex  \
0                   Allen, Miss Elisabeth Walton    1st  29.0  female
1                    Allison, Miss Helen Loraine    1st   2.0  female
2            Allison, Mr Hudson Joshua Creighton    1st  30.0    male
3  Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st  25.0  female

   Survived
0         1
1         0
2         0
3         0
```

[5]:
```
#Shape
print(dataframe.shape)

#Show statistics
print(dataframe.describe())
```

```
(1313, 5)
              Age     Survived
count  756.000000  1313.000000
mean    30.397989     0.342727
std     14.259049     0.474802
min      0.170000     0.000000
25%     21.000000     0.000000
50%     28.000000     0.000000
75%     39.000000     1.000000
max     71.000000     1.000000
```

[6]:
```
#Navigating DataFrames
```

```
#Selecting rows
dataframe.iloc[0] #Selects first row

dataframe.iloc[1:4] #Selecting 3 rows
dataframe.iloc[:4] #Selecting rows upto 4th row

#Set index to any other feature i.e Name
dataframe = dataframe.set_index(dataframe["Name"])

dataframe.loc["Allen, Miss Elisabeth Walton"]
```

[6]:
```
Name        Allen, Miss Elisabeth Walton
PClass                               1st
Age                                 29.0
Sex                               female
Survived                               1
Name: Allen, Miss Elisabeth Walton, dtype: object
```

[7]:
```
#loc is used when the index of the DataFrame is a label
#iloc works by looking for the position in the DataFrame.
```

[8]:
```
# Selecting Rows Based on Conditionals

#selecting only the female data
dataframe[dataframe["Sex"]=="female"].head(5)

#Females with age more than 65
dataframe[(dataframe["Sex"]=="female") & (dataframe["Age"]>=65)]

#Males who survived
dataframe[(dataframe["Sex"]=="male") & (dataframe["Survived"]==1)].head(5)
```

[8]:
```
                                                      Name PClass    Age  \
Name
Allison, Master Hudson Trevor  Allison, Master Hudson Trevor    1st   0.92
Anderson, Mr Harry                       Anderson, Mr Harry    1st  47.00
Barkworth, Mr Algernon H           Barkworth, Mr Algernon H    1st    NaN
Beckwith, Mr Richard Leord       Beckwith, Mr Richard Leord    1st  37.00
Behr, Mr Karl Howell                   Behr, Mr Karl Howell    1st  26.00

                                Sex  Survived
Name
Allison, Master Hudson Trevor  male         1
Anderson, Mr Harry             male         1
Barkworth, Mr Algernon H       male         1
Beckwith, Mr Richard Leord     male         1
Behr, Mr Karl Howell           male         1
```

[9]:
```
# Replacing Values in a DataFrame
#DataFrame.replace(a,b) -- replace a with b

dataframe["Survived"].replace(1,"alive").head(5)

dataframe["Sex"].replace("female","Woman").head(5)

dataframe["Sex"].replace(["female","male"],["F","M"]).head(5)

dataframe.replace(1,"one").head(6) #It replaces in whole data
```

[9]:
```
        Name  \
Name
Allen, Miss Elisabeth Walton                                    Allen, Miss
Elisabeth Walton
Allison, Miss Helen Loraine                                      Allison, Miss
Helen Loraine
Allison, Mr Hudson Joshua Creighton                      Allison, Mr Hudson
Joshua Creighton
Allison, Mrs Hudson JC (Bessie Waldo Daniels)  Allison, Mrs Hudson JC (Bessie
Waldo Daniels)
Allison, Master Hudson Trevor                                  Allison, Master
Hudson Trevor
Anderson, Mr Harry
Anderson, Mr Harry

                                              PClass   Age     Sex Survived
Name
Allen, Miss Elisabeth Walton                     1st  29.0  female      one
Allison, Miss Helen Loraine                      1st   2.0  female        0
Allison, Mr Hudson Joshua Creighton              1st  30.0    male        0
Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st  25.0  female        0
Allison, Master Hudson Trevor                    1st  0.92    male      one
Anderson, Mr Harry                               1st  47.0    male      one
```

[10]:
```
# Renaming Columns

dataframe.rename(columns={"PClass":"Passenger Class","Sex":"Gender"}).head(5)
#It takes a dictionary to replace names of one column or multiple columns
```

[10]:
```
        Name  \
Name
Allen, Miss Elisabeth Walton                                    Allen, Miss
Elisabeth Walton
Allison, Miss Helen Loraine                                      Allison, Miss
Helen Loraine
Allison, Mr Hudson Joshua Creighton                      Allison, Mr Hudson
```

```
Joshua Creighton
Allison, Mrs Hudson JC (Bessie Waldo Daniels)  Allison, Mrs Hudson JC (Bessie
Waldo Daniels)
Allison, Master Hudson Trevor                                  Allison, Master
Hudson Trevor

                                              Passenger Class    Age  Gender  \
Name
Allen, Miss Elisabeth Walton                              1st  29.00  female
Allison, Miss Helen Loraine                               1st   2.00  female
Allison, Mr Hudson Joshua Creighton                       1st  30.00    male
Allison, Mrs Hudson JC (Bessie Waldo Daniels)             1st  25.00  female
Allison, Master Hudson Trevor                             1st   0.92    male

                                               Survived
Name
Allen, Miss Elisabeth Walton                          1
Allison, Miss Helen Loraine                           0
Allison, Mr Hudson Joshua Creighton                   0
Allison, Mrs Hudson JC (Bessie Waldo Daniels)         0
Allison, Master Hudson Trevor                         1
```

[11]:
```python
import collections

column_names = collections.defaultdict(str)

for name in dataframe.columns:
    column_names[name]

print(column_names)
```

```
defaultdict(<class 'str'>, {'Name': '', 'PClass': '', 'Age': '', 'Sex': '',
'Survived': ''})
```

[12]:
```python
# Maximum, Minimum, Sum, Average and Count

print("Max: ",dataframe["Age"].max())
print("Min: ",dataframe["Age"].min())
print("Avg: ",dataframe["Age"].mean())
print("Sum: ",dataframe["Age"].sum())
print("Count: ",dataframe["Age"].count())
```

```
Max:  71.0
Min:  0.17
Avg:  30.397989417989418
Sum:  22980.88
Count:  756
```

[13]:
```
dataframe.count()
```

[13]:
```
Name        1313
PClass      1312
Age          756
Sex         1313
Survived    1313
dtype: int64
```

[14]:
```
# Unique Values
dataframe["PClass"].unique()

dataframe["PClass"].value_counts() #Freq table of unique values

dataframe["Sex"].value_counts()

dataframe["PClass"].nunique() #Give the count of unique values
```

[14]:
```
3
```

[15]:
```
# Missing values
dataframe[dataframe["Age"].isnull()].head(2)
```

[15]:
```
                                                      Name PClass  Age  \
Name
Aubert, Mrs Leontine Pauline  Aubert, Mrs Leontine Pauline    1st  NaN
Barkworth, Mr Algernon H          Barkworth, Mr Algernon H    1st  NaN

                                 Sex  Survived
Name
Aubert, Mrs Leontine Pauline  female         1
Barkworth, Mr Algernon H        male         1
```

[16]:
```
# Replacing values with NaN
import numpy as np
dataframe["Sex"].replace("male",np.nan)
```

[16]:
```
Name
Allen, Miss Elisabeth Walton                     female
Allison, Miss Helen Loraine                      female
Allison, Mr Hudson Joshua Creighton                 NaN
Allison, Mrs Hudson JC (Bessie Waldo Daniels)    female
Allison, Master Hudson Trevor                       NaN
                                                  …
Zakarian, Mr Artun                                  NaN
Zakarian, Mr Maprieder                              NaN
Zenni, Mr Philip                                    NaN
```

```
Lievens, Mr Rene                                    NaN
Zimmerman, Leo                                      NaN
Name: Sex, Length: 1313, dtype: object
```

[17]:
```
#Load Data, Set missing values
dataframe = pd.read_csv(url,na_values=[np.nan,"NONE",-999])
```

[18]:
```
# Deleting Columns

dataframe.drop("Age",axis=1).head(5)

dataframe.drop(["Age","Sex"],axis=1).head(5)

dataframe.drop(dataframe.columns[1],axis=1).head(5) #using index if names not␣
↪available
```

[18]:
```
                                       Name    Age     Sex  Survived
0                 Allen, Miss Elisabeth Walton  29.00  female         1
1                  Allison, Miss Helen Loraine   2.00  female         0
2          Allison, Mr Hudson Joshua Creighton  30.00    male         0
3  Allison, Mrs Hudson JC (Bessie Waldo Daniels)  25.00  female         0
4                Allison, Master Hudson Trevor   0.92    male         1
```

[19]:
```
# Deleting Rows

dataframe[dataframe["Sex"] != "female"].head(4)

dataframe[dataframe["Age"] >=18 ].head(4)

dataframe[dataframe.index != 0].head(3) #Using index
```

[19]:
```
                                       Name PClass   Age     Sex  \
1                  Allison, Miss Helen Loraine    1st   2.0  female
2          Allison, Mr Hudson Joshua Creighton    1st  30.0    male
3  Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st  25.0  female

   Survived
1         0
2         0
3         0
```

[20]:
```
# Dropping Duplicate rows
dataframe.drop_duplicates().head(4) #It will drop only when the both rows have␣
↪exact row values

dataframe.drop_duplicates(subset=["Sex"]) # only unique value will be obtained
dataframe.drop_duplicates(subset=["Sex"],keep="last") #keeps last unique
```

[20]:
```
                       Name PClass   Age     Sex  Survived
1307   Zabour, Miss Tamini    3rd   NaN  female         0
1312        Zimmerman, Leo    3rd  29.0    male         0
```

[21]:
```
# Grouping rows by values

dataframe.groupby("Sex").mean()

dataframe.groupby("Survived")['Name'].count()

dataframe.groupby("Survived").count()

dataframe.groupby(["Sex","Survived"])["Age"].count()
```

[21]:
```
Sex     Survived
female  0            71
        1           217
male    0           372
        1            96
Name: Age, dtype: int64
```

[22]:
```
# Group Rows by Time
#Create date range
time_index = pd.date_range("14/12/2012",periods=100000,freq="30S") # "30S"-30␣
 ↪secs frequency

#Create DataFrame
dataframe = pd.DataFrame(index=time_index)

#Add Columns
dataframe["Sale_Amount"] = np.random.randint(1,10,100000)

#Group rows by week, calculate sum per week
dataframe["Sale_Amount"].resample("W").sum()

dataframe["Sale_Amount"].resample("2W").mean() # 2 weeks mean

dataframe["Sale_Amount"].resample("M").count() #monthly count

#resample returns the label of the right "edge" (the last label) of the time␣
 ↪group.
#We can control this behavior using the label parameter:

dataframe["Sale_Amount"].resample("M",label="left").count()
```

[22]:
```
2012-11-30    51840
2012-12-31    48160
Freq: M, Name: Sale_Amount, dtype: int64
```

[23]:
```
# Looping over a column

url = "https://osf.io/aupb4/download"

dataframe = pd.read_csv(url)

for name in dataframe["Name"][0:5]:
    print(name.upper())

#List comprehension can be used as
print([name.upper() for name in dataframe["Name"][0:5]])
```

```
ALLEN, MISS ELISABETH WALTON
ALLISON, MISS HELEN LORAINE
ALLISON, MR HUDSON JOSHUA CREIGHTON
ALLISON, MRS HUDSON JC (BESSIE WALDO DANIELS)
ALLISON, MASTER HUDSON TREVOR
['ALLEN, MISS ELISABETH WALTON', 'ALLISON, MISS HELEN LORAINE', 'ALLISON, MR
HUDSON JOSHUA CREIGHTON', 'ALLISON, MRS HUDSON JC (BESSIE WALDO DANIELS)',
'ALLISON, MASTER HUDSON TREVOR']
```

[24]:
```
# Applying a Function over all elements in a cloumn

#Creating a function
def uppercase(x):
    return x.upper()

#apply function to column
dataframe["Name"].apply(uppercase)[0:5]
```

[24]:
```
0                      ALLEN, MISS ELISABETH WALTON
1                       ALLISON, MISS HELEN LORAINE
2               ALLISON, MR HUDSON JOSHUA CREIGHTON
3     ALLISON, MRS HUDSON JC (BESSIE WALDO DANIELS)
4                     ALLISON, MASTER HUDSON TREVOR
Name: Name, dtype: object
```

[25]:
```
# Applying a Function to groups

'''By combining groupby and apply we can calculate cus-
tom statistics or apply any function to each group separately.'''

dataframe.groupby("Sex").apply(lambda x:x.count())
```

```
[25]:         Name  PClass  Age  Sex  Survived
      Sex
      female   462     462  288  462       462
      male     851     850  468  851       851
```

```
[26]: # Concatenating DataFrames

      data1 = {"id":["1","2","3"],
              "name":["sd","ds","ss"]}
      dataframe1 = pd.DataFrame(data1,columns=["id","name"])

      data2 = {"id":["4","5","6"],
              "name":["fd","df","ff"]}
      dataframe2 = pd.DataFrame(data2,columns=["id","name"])

      pd.concat([dataframe1,dataframe2],axis=0)

      pd.concat([dataframe1,dataframe2],axis=1)

      # Add Series to Dataframe
      row = pd.Series(["5","huihedf"],index=["id","name"])

      dataframe1.append(row,ignore_index=True)
```

```
[26]:   id     name
      0  1       sd
      1  2       ds
      2  3       ss
      3  5  huihedf
```

```
[27]: # Merging DataFrames
      import pandas as pd
      employees = {"employee_id":['1','2','3','4'],
                  'name':["A","B","C","D"]}
      dataframe_employees = pd.DataFrame(employees,columns=["employee_id","name"])

      sales = {"employee_id":["3","4","5","6"],
              "total_sales":[1323,3254,2343,424]}
      dataframe_sales = pd.DataFrame(sales,columns=["employee_id","total_sales"])

      #Merge DataFrames
      pd.merge(dataframe_employees,dataframe_sales,on="employee_id")
      # Defaults to inner join
```

```
[27]:   employee_id name  total_sales
      0           3    C         1323
      1           4    D         3254
```

```
[28]: # Outer join
      pd.merge(dataframe_employees,dataframe_sales,on="employee_id",how="outer")
```

```
[28]:   employee_id name  total_sales
      0            1    A          NaN
      1            2    B          NaN
      2            3    C       1323.0
      3            4    D       3254.0
      4            5  NaN       2343.0
      5            6  NaN        424.0
```

```
[29]: # Left join
      pd.merge(dataframe_employees,dataframe_sales,on="employee_id",how="left")
```

```
[29]:   employee_id name  total_sales
      0            1    A          NaN
      1            2    B          NaN
      2            3    C       1323.0
      3            4    D       3254.0
```

```
[30]: # Right join
      pd.merge(dataframe_employees,dataframe_sales,on="employee_id",how="right")
```

```
[30]:   employee_id name  total_sales
      0            3    C         1323
      1            4    D         3254
      2            5  NaN         2343
      3            6  NaN          424
```

```
[31]: '''If instead of merging on two columns we want to merge on the indexes of each␣
       ↪Data-
      Frame, we can replace the left_on and right_on parameters with right_index=True
      and left_index=True .'''

      pd.merge(dataframe_employees,
      dataframe_sales,
      left_on='employee_id',
      right_on='employee_id')
```

```
[31]:   employee_id name  total_sales
      0            3    C         1323
      1            4    D         3254
```

```
[32]: #Joins
      '''Inner
      Return only the rows that match in both DataFrames (e.g., return any row with
      an employee_id value appearing in both dataframe_employees and data
      frame_sales ).
      Outer
      Return all rows in both DataFrames. If a row exists in one DataFrame but not in
      the other DataFrame, fill NaN values for the missing values (e.g., return all␣
       ↪rows
      in both employee_id and dataframe_sales ).
      Left
      Return all rows from the left DataFrame but only rows from the right DataFrame
      that matched with the left DataFrame. Fill NaN values for the missing values (e.
       ↪g.,
      return all rows from dataframe_employees but only rows from data
      frame_sales that have a value for employee_id that appears in data
      frame_employees ).
      Right
      Return all rows from the right DataFrame but only rows from the left DataFrame
      that matched with the right DataFrame. Fill NaN values for the missing values
      (e.g., return all rows from dataframe_sales but only rows from data␣
       ↪frame_employees that have a value for employee_id that appears in data
      frame_sales ).'''
```

[32]:
```
'Inner\nReturn only the rows that match in both DataFrames (e.g., return any row
with\nan employee_id value appearing in both dataframe_employees and
data\nframe_sales ).\nOuter\nReturn all rows in both DataFrames. If a row exists
in one DataFrame but not in\nthe other DataFrame, fill NaN values for the
missing values (e.g., return all rows\nin both employee_id and dataframe_sales
).\nLeft\nReturn all rows from the left DataFrame but only rows from the right
DataFrame\nthat matched with the left DataFrame. Fill NaN values for the missing
values (e.g.,\nreturn all rows from dataframe_employees but only rows from
data\nframe_sales that have a value for employee_id that appears in
data\nframe_employees ).\nRight\nReturn all rows from the right DataFrame but
only rows from the left DataFrame\nthat matched with the right DataFrame. Fill
NaN values for the missing values\n(e.g., return all rows from dataframe_sales
but only rows from data frame_employees that have a value for employee_id that
appears in data\nframe_sales ).'
```