Lab02-AcquiringAndPrepDataQ

October 27, 2023

SUBMISSION INFORMATION:

Student Name: Yousef Ibrahim Gomaa Mahmoud

Group: AID 1

Section: 3

ID: 320210207

Email: yousef.gomaa@ejust.edu.eg

Acquire and Prepare Data

Importing the libraries

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

Importing the dataset

[ ]:
```
#NBA players
df=pd.read_csv("Playerdata.csv")
```

[ ]:
```
df.tail(10)
```

[ ]:

	Name	Team	Position	Age	Weight	College	Salary
447	Rudy Gobert	Utah Jazz	C	(23+0j)	245	NaN	1175880.0
448	Gordon Hayward	Utah Jazz	SF	(26+0j)	226	Butler	15409570.0
449	Rodney Hood	Utah Jazz	SG	(23+0j)	206	Duke	1348440.0
450	Joe Ingles	Utah Jazz	SF	(28+0j)	226	NaN	2050000.0
451	Chris Johnson	Utah Jazz	SF	(26+0j)	206	Dayton	981348.0
452	Trey Lyles	Utah Jazz	PF	(20+0j)	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	PG	(26+0j)	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	PG	(24+0j)	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	C	(26+0j)	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	C	(26+0j)	231	Kansas	947276.0

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

[ ]:

	Weight	Salary
count	457.000000	4.460000e+02

```
mean   221.522976  4.842684e+06
std     26.368343  5.229238e+06
min    161.000000  3.088800e+04
25%    200.000000  1.044792e+06
50%    220.000000  2.839073e+06
75%    240.000000  6.500000e+06
max    307.000000  2.500000e+07
```

[ ]:
```
df.head()#try df.head(number)
```

[ ]:
```
              Name            Team Position      Age  Weight            College  \
0  Avery Bradley  Boston Celtics       PG  (25+0j)     180              Texas
1    Jae Crowder  Boston Celtics       SF  (25+0j)     235          Marquette
2   John Holland  Boston Celtics       SG  (27+0j)     205  Boston University
3    R.J. Hunter  Boston Celtics       SG  (22+0j)     185      Georgia State
4  Jonas Jerebko  Boston Celtics       PF  (29+0j)     231                NaN

      Salary
0  7730337.0
1  6796117.0
2        NaN
3  1148640.0
4  5000000.0
```

Check missing Values

[ ]:
```
df.isna().sum() # Conclusion is in the markdown cell below.
```

[ ]:
```
Name         0
Team         0
Position     0
Age          0
Weight       0
College     84
Salary      11
dtype: int64
```

Explain what you found?

Since "isna()" method returns true for every null/na (missing) value in the dataset, otherwise false, we count the sum of the nulls in each column. According to the output, it is deduced that feature "College" has 84 nulls, meaning that 84 of the entries have missing college info or has not went to college. As for "Salary", there are 11 missing values.

Write code for handeling any missing

[ ]:
```
df.fillna(df.mean(numeric_only=True), inplace=True) # Fill numerical data with␣
 ↪average. (i.e: 'Salary')
df.isna().sum()
```

[ ]:
```
Name          0
Team          0
Position      0
Age           0
Weight        0
College      84
Salary        0
dtype: int64
```

[ ]:
```
df.dropna(axis=0, inplace=True) # Drop rest of records (categorical) with␣
 ↪missing values and update the dataframe itself with its new values.
df.isna().sum()
```

[ ]:
```
Name        0
Team        0
Position    0
Age         0
Weight      0
College     0
Salary      0
dtype: int64
```

[ ]:
```
df.info() # Age and Salary do not have correct data types.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 373 entries, 0 to 456
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      373 non-null    object
 1   Team      373 non-null    object
 2   Position  373 non-null    object
 3   Age       373 non-null    object
 4   Weight    373 non-null    int64
 5   College   373 non-null    object
 6   Salary    373 non-null    float64
dtypes: float64(1), int64(1), object(5)
memory usage: 23.3+ KB
```

you see that Age & Weight & Salary not accurate data types , write your code to handle them

[ ]:
```
df.Age = df['Age'].str.lstrip('(').str.rstrip('+0j)') # Trim left of the string␣
 ↪if '(' is found and trim the right of the string if '+0j)' is found.
# |||                                        |||
# vvv Casting data types to correct ones vvv
df.Age.astype('int32')
df.Salary.astype('int64') # Larger integer values for Salary.
```

```
df.Weight.astype('float64') # Normally, weight has fractions either in pounds␣
 ↪or kilograms but rounded. (optional)
```

[ ]:
```
0      180.0
1      235.0
2      205.0
3      185.0
6      235.0
       …
449    206.0
451    206.0
452    234.0
453    203.0
456    231.0
Name: Weight, Length: 373, dtype: float64
```

Display Age and Salary as a dataframe

[ ]:
```
AaS = df[['Age','Salary']] # Only selects Age and Salary.
AaS.sample(5)
```

[ ]:
```
     Age      Salary
21    26    134215.0
203   28    211744.0
168   24  16407501.0
334    3   5675000.0
404   25   1938840.0
```

Filter data according to Salary below 60000

[ ]:
```
AaS[AaS['Salary'] < 60000]
```

[ ]:
```
     Age   Salary
130   25  55722.0
291   27  55722.0
```

Remove the rows that satisfy the condition

[ ]:
```
AaS = AaS.drop(index=AaS[AaS['Salary'] < 60000].index) # Drops rows whose␣
 ↪Salary values are less than 60000.
```

[ ]:
```
AaS[AaS['Salary'] < 60000] # Removed successfully.
```

[ ]:
```
Empty DataFrame
Columns: [Age, Salary]
Index: []
```

Encoding categorical data - Example

[ ]:
```
df['Position'].unique()
```

[ ]:
```
array(['PG', 'SF', 'SG', 'PF', 'C'], dtype=object)
```

[ ]:
```
#example categorical - ordinal (with order)
#PG: 1 , SG:2, SF:3, PF: 4 , C:5
ser1=pd.Series([1,2,3,4,5],index=['PG','SG','SF','PF','C'])
df['Position']=df['Position'].replace(ser1)
```