data_wrangling_Ass_07

March 15, 2022

1 Data Wrangling

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

[ ]:
```
kashti = sns.load_dataset('titanic')
ks1 = kashti
ks2 = kashti
```

[ ]:
```
kashti.head()
```

[ ]:
```
   Unnamed: 0  survived  pclass     sex   age  sibsp  parch     fare embarked  \
0           0         0       3    male  22.0      1      0   7.2500        S
1           1         1       1  female  38.0      1      0  71.2833        C
2           2         1       3  female  26.0      0      0   7.9250        S
3           3         1       1  female  35.0      1      0  53.1000        S
4           4         0       3    male  35.0      0      0   8.0500        S

   class    who  adult_male deck  embark_town alive  alone
0  Third    man        True  NaN  Southampton    no  False
1  First  woman       False    C    Cherbourg   yes  False
2  Third  woman       False  NaN  Southampton   yes   True
3  First  woman       False    C  Southampton   yes  False
4  Third    man        True  NaN  Southampton    no   True
```

[ ]:
```
(kashti['age']+1).head(10)
```

[ ]:
```
0    23.0
1    39.0
2    27.0
3    36.0
4    36.0
5     NaN
6    55.0
7     3.0
8    28.0
9    15.0
```

```
Name: age, dtype: float64
```

1.1 Dealing with the missing Values

- In a data set missing values are N/A, NAN, or zero
- Remove the missing value variable
- Replace the missing Value :
  1. How?
     - Average Accuray
     - Frequency or mode replacement
     - Replace Based on other functions
     - ML Algos can also used
     - Ignore the missing Values
  2. Why?
     - to avoid data loss
     - Less accurate

[ ]:
```
# where exactly missing values are
kashti.isnull().sum()
```

[ ]:
```
Unnamed: 0       0
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

[ ]:
```
kashti.shape
```

[ ]:
```
(891, 16)
```

[ ]:
```
# To drop the NaN values
kashti.dropna(subset=['deck'],axis=0, inplace=True)
```

[ ]:
```
kashti.isnull().sum()
```

[ ]:
```
Unnamed: 0      0
survived        0
pclass          0
sex             0
age            19
sibsp           0
parch           0
fare            0
embarked        2
class           0
who             0
adult_male      0
deck            0
embark_town     2
alive           0
alone           0
dtype: int64
```

[ ]:
```
# Remove all the missing values as whole
kashti.dropna()
kashti.dropna().isnull().sum()
```

[ ]:
```
Unnamed: 0     0
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

[ ]:
```
kashti.shape
```

[ ]:
```
(203, 16)
```

[ ]:
```
ks1.isnull().sum()
```

[ ]:
```
Unnamed: 0      0
survived        0
pclass          0
sex             0
age            19
sibsp           0
parch           0
fare            0
embarked        2
class           0
who             0
adult_male      0
deck            0
embark_town     2
alive           0
alone           0
dtype: int64
```

[ ]:
```
mean = ks1['age'].mean()
```

[ ]:
```
# Replacing the missing values with the mean
ks1['age'] = ks1['age'].replace(np.nan, mean)
```

[ ]:
```
ks1.dropna(subset=['deck'],axis=0, inplace=True)
```

[ ]:
```
ks1.dropna(subset=['embarked'],axis=0, inplace=True)
```

[ ]:
```
ks1.isnull().sum()
```

[ ]:
```
Unnamed: 0     0
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

[ ]:
```
ks1.shape
```

[ ]:
```
(201, 16)
```

1.2 Data Formating

- Easy to gather
- Easy to understand

[ ]:
```
ks1.dtypes
```

[ ]:
```
Unnamed: 0        int64
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

[ ]:
```
ks1['survived'] = ks1['survived'].astype('float64')
ks1.dtypes
```

[ ]:
```
Unnamed: 0        int64
survived        float64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
```

```
dtype: object
```

```
[ ]: # convert the age into days
ks1['age'] = ks1['age']*356
```

```
[ ]: ks1.head(5)
```

```
[ ]:     Unnamed: 0  survived  pclass     sex      age  sibsp  parch     fare  \
1              1       1.0       1  female  13528.0      1      0  71.2833
3              3       1.0       1  female  12460.0      1      0  53.1000
6              6       0.0       1    male  19224.0      0      0  51.8625
10            10       1.0       3  female   1424.0      1      1  16.7000
11            11       1.0       1  female  20648.0      0      0  26.5500

    embarked  class    who  adult_male deck  embark_town alive  alone
1          C  First  woman       False    C    Cherbourg   yes  False
3          S  First  woman       False    C  Southampton   yes  False
6          S  First    man        True    E  Southampton    no   True
10         S  Third  child       False    G  Southampton   yes  False
11         S  First  woman       False    C  Southampton   yes   True
```

```
[ ]: ks1.rename(columns={"age": "age in days"}, inplace=True)
ks1.head(5)
```

```
[ ]:     Unnamed: 0  survived  pclass     sex  age in days  sibsp  parch     fare  \
1              1       1.0       1  female       13528.0      1      0  71.2833
3              3       1.0       1  female       12460.0      1      0  53.1000
6              6       0.0       1    male       19224.0      0      0  51.8625
10            10       1.0       3  female        1424.0      1      1  16.7000
11            11       1.0       1  female       20648.0      0      0  26.5500

    embarked  class    who  adult_male deck  embark_town alive  alone
1          C  First  woman       False    C    Cherbourg   yes  False
3          S  First  woman       False    C  Southampton   yes  False
6          S  First    man        True    E  Southampton    no   True
10         S  Third  child       False    G  Southampton   yes  False
11         S  First  woman       False    C  Southampton   yes   True
```

1.3 Data Normalization

- Make the data Uniform
- Make sure that the whole data have the same impact
- Aik Machli smandar me or ek Jar me
- To minimize the computaional complexity

```
[ ]: ks3 = ks1[['age in days', 'fare']]
ks3.head(5)
```

[ ]:

	age in days	fare
1	13528.0	71.2833
3	12460.0	53.1000
6	19224.0	51.8625
10	1424.0	16.7000
11	20648.0	26.5500

[ ]:
```
ks3['fare'] = ks3['fare']/ks3['fare'].max()
ks3.head()
```

```
C:\Users\Sartaj\AppData\Local\Temp\ipykernel_18296\239786220.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ks3['fare'] = ks3['fare']/ks3['fare'].max()
```

[ ]:

	age in days	fare
1	13528.0	0.139136
3	12460.0	0.103644
6	19224.0	0.101229
10	1424.0	0.032596
11	20648.0	0.051822

[ ]:
```
# Simple feature scaling
ks3['age in days'] = ks3['age in days']/ks3['age in days'].max()
ks3.head()
```

```
C:\Users\Sartaj\AppData\Local\Temp\ipykernel_18296\4274444637.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ks3['age in days'] = ks3['age in days']/ks3['age in days'].max()
```

[ ]:

	age in days	fare
1	0.4750	0.139136
3	0.4375	0.103644
6	0.6750	0.101229
10	0.0500	0.032596
11	0.7250	0.051822

[ ]:
```
# mean max method
```

```
ks3['fare'] = (ks3['fare']-ks3['fare'].min())/(ks3['fare'].max()-ks3['fare'].
 ↪min())
ks3.head()
```

```
C:\Users\Sartaj\AppData\Local\Temp\ipykernel_18296\2673459693.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ks3['fare'] =
(ks3['fare']-ks3['fare'].min())/(ks3['fare'].max()-ks3['fare'].min())
```

[ ]:
```
    age in days      fare
1        0.4750  0.139136
3        0.4375  0.103644
6        0.6750  0.101229
10       0.0500  0.032596
11       0.7250  0.051822
```

[ ]:
```
# z_score
ks3['fare'] = (ks3['fare']-ks3['fare'].mean())/ks3['fare'].std()
ks3.head()
```

```
C:\Users\Sartaj\AppData\Local\Temp\ipykernel_18296\130998099.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ks3['fare'] = (ks3['fare']-ks3['fare'].mean())/ks3['fare'].std()
```

[ ]:
```
    age in days      fare
1        0.4750 -0.067057
3        0.4375 -0.309853
6        0.6750 -0.326377
10       0.0500 -0.795891
11       0.7250 -0.664367
```

[ ]:
```
# Log transformation
ks = sns.load_dataset('titanic')
ks.head(5)
```

[ ]:
```
   Unnamed: 0  survived  pclass     sex   age  sibsp  parch     fare embarked  \
0           0         0       3    male  22.0      1      0   7.2500        S
1           1         1       1  female  38.0      1      0  71.2833        C
```

```
2           2         1       3  female  26.0      0      0   7.9250        S
3           3         1       1  female  35.0      1      0  53.1000        S
4           4         0       3    male  35.0      0      0   8.0500        S

   class    who  adult_male deck  embark_town alive  alone
0  Third    man        True  NaN  Southampton    no  False
1  First  woman       False    C    Cherbourg   yes  False
2  Third  woman       False  NaN  Southampton   yes   True
3  First  woman       False    C  Southampton   yes  False
4  Third    man        True  NaN  Southampton    no   True
```

[ ]:
```
ks['fare'] = np.log(ks['fare'])
ks.head()
```

```
C:\Users\Sartaj\AppData\Local\Programs\Python\Python39\lib\site-
packages\pandas\core\arraylike.py:358: RuntimeWarning: divide by zero
encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

[ ]:
```
   Unnamed: 0  survived  pclass     sex   age  sibsp  parch      fare  \
0           0         0       3    male  22.0      1      0  1.981001
1           1         1       1  female  38.0      1      0  4.266662
2           2         1       3  female  26.0      0      0  2.070022
3           3         1       1  female  35.0      1      0  3.972177
4           4         0       3    male  35.0      0      0  2.085672

  embarked  class    who  adult_male deck  embark_town alive  alone
0        S  Third    man        True  NaN  Southampton    no  False
1        C  First  woman       False    C    Cherbourg   yes  False
2        S  Third  woman       False  NaN  Southampton   yes   True
3        S  First  woman       False    C  Southampton   yes  False
4        S  Third    man        True  NaN  Southampton    no   True
```

[ ]:
```
ks.tail(5)
```

[ ]:
```
     Unnamed: 0  survived  pclass     sex   age  sibsp  parch      fare  \
886         886         0       2    male  27.0      0      0  2.564949
887         887         1       1  female  19.0      0      0  3.401197
888         888         0       3  female   NaN      1      2  3.154870
889         889         1       1    male  26.0      0      0  3.401197
890         890         0       3    male  32.0      0      0  2.047693

    embarked   class    who  adult_male deck  embark_town alive  alone
886        S  Second    man        True  NaN  Southampton    no   True
887        S   First  woman       False    B  Southampton   yes   True
888        S   Third  woman       False  NaN  Southampton    no  False
889        C   First    man        True    C    Cherbourg   yes   True
890        Q   Third    man        True  NaN   Queenstown    no   True
```

[ ]:
```
min(ks['age'])
```

[ ]: 0.42

[ ]:
```
max(ks['age'])
```

[ ]: 80.0

[ ]:
```
ks = ks.dropna()
```

[ ]:
```
k = sns.load_dataset('titanic')
k = k.dropna()
```

[ ]:
```
k.tail()
```

[ ]:
```
     Unnamed: 0  survived  pclass     sex   age  sibsp  parch     fare  \
871         871         1       1  female  47.0      1      1  52.5542
872         872         0       1    male  33.0      0      0   5.0000
879         879         1       1  female  56.0      0      1  83.1583
887         887         1       1  female  19.0      0      0  30.0000
889         889         1       1    male  26.0      0      0  30.0000

    embarked  class    who  adult_male deck  embark_town alive  alone
871        S  First  woman       False    D  Southampton   yes  False
872        S  First    man        True    B  Southampton    no   True
879        C  First  woman       False    C    Cherbourg   yes  False
887        S  First  woman       False    B  Southampton   yes   True
889        C  First    man        True    C    Cherbourg   yes   True
```

[ ]:
```
k['age']
```

[ ]:
```
1      38.0
3      35.0
6      54.0
10      4.0
11     58.0
       …
871    47.0
872    33.0
879    56.0
887    19.0
889    26.0
Name: age, Length: 182, dtype: float64
```

[ ]:
```
# Binning
k['age'] = pd.cut(k['age'].astype(int),5)
k['age']
```

[ ]:
```
1       (32.0, 48.0]
3       (32.0, 48.0]
6       (48.0, 64.0]
10     (-0.08, 16.0]
11      (48.0, 64.0]
           ...
871     (32.0, 48.0]
872     (32.0, 48.0]
879     (48.0, 64.0]
887     (16.0, 32.0]
889     (16.0, 32.0]
Name: age, Length: 182, dtype: category
Categories (5, interval[float64]): [(-0.08, 16.0] < (16.0, 32.0] < (32.0, 48.0]
< (48.0, 64.0] < (64.0, 80.0]]
```

[ ]:
```
pd.get_dummies(ks['sex'])
ks.head()
```

[ ]:
```
     Unnamed: 0  survived  pclass     sex   age  sibsp  parch      fare  \
1             1         1       1  female  38.0      1      0  4.266662
3             3         1       1  female  35.0      1      0  3.972177
6             6         0       1    male  54.0      0      0  3.948596
10           10         1       3  female   4.0      1      1  2.815409
11           11         1       1  female  58.0      0      0  3.279030

   embarked  class    who  adult_male deck  embark_town alive  alone
1         C  First  woman       False    C    Cherbourg   yes  False
3         S  First  woman       False    C  Southampton   yes  False
6         S  First    man        True    E  Southampton    no   True
10        S  Third  child       False    G  Southampton   yes  False
11        S  First  woman       False    C  Southampton   yes   True
```

[ ]:
```
dum = pd.get_dummies(ks['sex'])
dum
```

[ ]:
```
     female  male
1         1     0
3         1     0
6         0     1
10        1     0
11        1     0
..      ...   ...
871       1     0
872       0     1
879       1     0
887       1     0
889       0     1
```

```
[182 rows x 2 columns]
```

[ ]:

[ ]: