Data Wrangling

January 18, 2022

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

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

[ ]:
```
kashti
```

[ ]:
```
kashti.dtypes
```

[ ]:
```
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
```

1 Binning

- Grouping of values into smaller number of values (bins)
- convert numeric to categories [child, young, old etc]
- to have better understanding of groups
  - low vs mid vs high prices

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

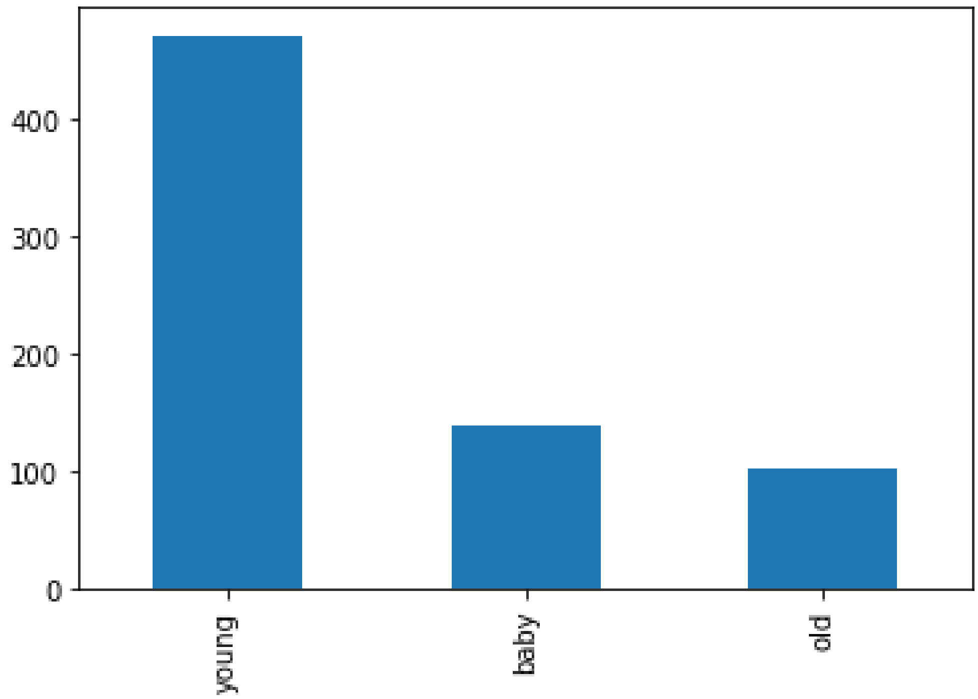
[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: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

[ ]:
```python
#use drop.na method
# print(kashti.shape)
kashti.dropna(subset=['age'], axis=0, inplace=True)
#inplace will replace changes in orginal data
```

[ ]:
```python
bins= [0,18,45,90]
labels= ['baby','young','old']
kashti['age_bin']=pd.cut(kashti['age'],bins,labels)
print(pd.value_counts(["kashti_bin"], sort=False))
kashti['categories']= pd.cut(kashti['age'],bins,labels=labels)
kashti['categories'].value_counts().plot(kind='bar')
```

```
kashti_bin    1
dtype: int64
```

[ ]: <AxesSubplot:>



[ ]:
```
#simple operations (math operations)
(kashti['age']+1).head()
```

[ ]:
```
0    23.0
1    39.0
2    27.0
3    36.0
4    36.0
Name: age, dtype: float64
```

2 Dealing with missing values

- In data set missing values are either empty or NAN ## Steps 1- try to download again 2- remove that row or column (if not effecting dataset) 3- Replace based on other functions 4- ML algorithm can also be used 5- Leave it like that 6- Frequency or MODE replacement ## Why 1- its better because no data lose 2- effect accuracy

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

[ ]:
```
survived        0
pclass          0
sex             0
age             0
sibsp           0
```

```
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           530
embark_town      2
alive            0
alone            0
age_bin          0
categories       0
dtype: int64
```

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

[ ]:
```
survived         0
pclass           0
sex              0
age              0
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           530
embark_town      2
alive            0
alone            0
age_bin          0
categories       0
dtype: int64
```

[ ]:
```
# to drop na values from all the data
kashti.dropna()
# to update the main dataframe
kashti= kashti.dropna()
kashti.isnull().sum() #remove na from whole
```

[ ]:
```
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
age_bin         0
categories      0
dtype: int64
```

[ ]:
```
kashti.shape #look if data is enough now
```

[ ]:
```
(182, 17)
```

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

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

3 Replacing missing values

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

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

[ ]:
```
<bound method NDFrame._add_numeric_operations.<locals>.mean of 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       …
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64>
```

[ ]:
```
# finding mean (average)
mean=ks1['deck'].mean
mean
```

[ ]:
```
<bound method NDFrame._add_numeric_operations.<locals>.mean of 0      NaN
1        C
2      NaN
3        C
4      NaN
      …
886    NaN
887      B
888    NaN
889      C
890    NaN
Name: deck, Length: 891, dtype: category
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']>
```

[ ]:
```
ks1['age']=ks1['age'].replace(np.nan, mean)
```

[ ]:
```
# use this method to convert datatypes from one to other
ks1['survived']= ks1 ['survived'].astype("int64")
ks1.dtypes
```

[ ]:
```
survived          int64
pclass            int64
sex              object
age              object
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
```

[ ]:
```
# replace na values of deck with mean
ks1['deck']=ks1['deck'].replace(np.nan, mean)
```

[ ]:
```
#use drop.na method for embark town
# print(ks1.shape)
ks1.dropna(subset=['embark_town'], axis=0, inplace=True)
#inplace will replace changes in orginal data
```

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

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

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

4 Data formatting

- one standard
- Ensure data is consistent and understandable
- easy to gather
- Easy to work with

[ ]:
```
# know the data type and convert it into the known type
kashti.dtypes
```

[ ]:
```
survived         int64
pclass           int64
sex             object
age             object
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
```

[ ]:
```
# use this method to convert datatypes from one to other
ks1['survived']= ks1 ['survived'].astype("int64")
ks1.dtypes
```

[ ]:
```
survived            int64
pclass              int64
sex                object
age                object
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 age into days
ks=sns.load_dataset('titanic')
ks['age']=ks['age']*365
ks.head(10)
```

[ ]:
```
   survived  pclass     sex      age  sibsp  parch     fare embarked  class  \
0         0       3    male   8030.0      1      0   7.2500        S  Third
1         1       1  female  13870.0      1      0  71.2833        C  First
2         1       3  female   9490.0      0      0   7.9250        S  Third
3         1       1  female  12775.0      1      0  53.1000        S  First
4         0       3    male  12775.0      0      0   8.0500        S  Third
5         0       3    male      NaN      0      0   8.4583        Q  Third
6         0       1    male  19710.0      0      0  51.8625        S  First
7         0       3    male    730.0      3      1  21.0750        S  Third
8         1       3  female   9855.0      0      2  11.1333        S  Third
```

```
9         1        2  female   5110.0      1       0  30.0708        C  Second

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
5    man        True  NaN   Queenstown    no   True
6    man        True    E  Southampton    no   True
7  child       False  NaN  Southampton    no  False
8  woman       False  NaN  Southampton   yes  False
9  child       False  NaN    Cherbourg   yes  False
```

[ ]:
```
ks.dtypes
```

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

[ ]:
```
# to remove zeros from age values
# can convert from float to integer
ks['age']= ks['age'].astype("int")
ks.dtypes
```

[ ]:
```
survived          int64
pclass            int64
sex              object
age               int32
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
```

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

[ ]:
```
   survived  pclass     sex    age  sibsp  parch     fare embarked  class  \
0         0       3    male   8030      1      0   7.2500        S  Third
1         1       1  female  13870      1      0  71.2833        C  First
2         1       3  female   9490      0      0   7.9250        S  Third
3         1       1  female  12775      1      0  53.1000        S  First
```

```
4         0       3    male  12775      0      0   8.0500        S  Third

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

[ ]:
```
# after conversion always remname the column according to operation on that
ks.rename(columns={"age":"age in days"}, inplace=True)
ks.head()
```

[ ]:
```
   survived  pclass     sex  age in days  sibsp  parch     fare embarked  \
0         0       3    male         8030      1      0   7.2500        S
1         1       1  female        13870      1      0  71.2833        C
2         1       3  female         9490      0      0   7.9250        S
3         1       1  female        12775      1      0  53.1000        S
4         0       3    male        12775      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
```

4.0.1 Data normalization

- uniform the data
- making sure they have same impact
- also for computational reasons

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

[ ]:
```
ks= ks[['age in days','fare']]
ks.head()
```

[ ]:
```
   age in days     fare
0         8030   7.2500
1        13870  71.2833
2         9490   7.9250
3        12775  53.1000
4        12775   8.0500
```

4.0.2 Above data has very wide range so need to be normalize

5 Method of normalization

- simple feature scaling
  - x(new)=x(old)/x(max)
- min-max method
- Z-score (Standart score) -3 to +3
- Log transformation

[ ]:
```
# simple scaling method
ks['fare']= ks['fare']/ks['fare'].max()
ks.head()
```

[ ]:

	age in days	fare
0	8030	0.014151
1	13870	0.139136
2	9490	0.015469
3	12775	0.103644
4	12775	0.015713

[ ]:
```
# simple scaling method
ks['age in days']= ks['age in days']/ks['age in days'].max()
ks.head()
```

[ ]:

	age in days	fare
0	0.2750	0.014151
1	0.4750	0.139136
2	0.3250	0.015469
3	0.4375	0.103644
4	0.4375	0.015713

[ ]:
```
# min-max method
ks1['fare']=(ks1['fare']-ks1['fare'].min())/(ks1['fare'].max()-ks1['fare'].
 ↪min())
```

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

[ ]:
```
ks[['age in days','fare']]
```

[ ]:

	age in days	fare
0	0.2750	0.014151
1	0.4750	0.139136
2	0.3250	0.015469
3	0.4375	0.103644
4	0.4375	0.015713
..	…	…
885	0.4875	0.056848

```
886         0.3375  0.025374
887         0.2375  0.058556
889         0.3250  0.058556
890         0.4000  0.015127

[714 rows x 2 columns]
```

[ ]:
```
# Z-score (standard score)
ks4['fare']= (ks4['fare'].mean())/ks4['fare'].std()
ks4.head()
```

[ ]:
```
    age in days          fare
1        0.4750  3.641179e+15
3        0.4375  3.641179e+15
6        0.6750  3.641179e+15
10       0.0500  3.641179e+15
11       0.7250  3.641179e+15
```

[ ]:
```
#log transformation
ks=sns.load_dataset('titanic')
ks.head(3)
```

[ ]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third

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

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

```
C:\Anaconda\lib\site-packages\pandas\core\arraylike.py:364: RuntimeWarning:
divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

[ ]:
```
   survived  pclass   age  sibsp  parch      fare embarked  class    who  \
0         0       3  22.0      1      0  1.981001        S  Third    man
1         1       1  38.0      1      0  4.266662        C  First  woman
2         1       3  26.0      0      0  2.070022        S  Third  woman
3         1       1  35.0      1      0  3.972177        S  First  woman
4         0       3  35.0      0      0  2.085672        S  Third    man

   adult_male deck  embark_town alive  alone  female  male
0        True  NaN  Southampton    no  False       0     1
```

```
1       False    C    Cherbourg   yes  False         1      0
2       False  NaN  Southampton   yes   True         1      0
3       False    C  Southampton   yes  False         1      0
4        True  NaN  Southampton    no   True         0      1
```

[ ]:
```
ks['age'].head()
```

[ ]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

5.0.1 converting in dummies values

- easy to use for computation
- male, female (0,1)

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

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

[ ]:
```
     female  male
0         0     1
1         1     0
2         1     0
3         1     0
4         0     1
..      …    …
886       0     1
887       1     0
888       1     0
889       0     1
890       0     1

[891 rows x 2 columns]
```

[ ]:
```
# Get one hot encoding of columns 'Sex'
one_hot = pd.get_dummies(ks['sex'])
# Drop column as it is now encoded
ks = ks.drop('sex',axis = 1)
# Join the encoded df
ks = ks.join(one_hot)
ks
```

[ ]:
```
     survived  pclass   age  sibsp  parch     fare embarked  class    who  \
0           0       3  22.0      1      0   7.2500        S  Third    man
```

```
1            1       1  38.0      1      0  71.2833        C   First  woman
2            1       3  26.0      0      0   7.9250        S   Third  woman
3            1       1  35.0      1      0  53.1000        S   First  woman
4            0       3  35.0      0      0   8.0500        S   Third    man
..         …     …   …    …    …      …      …      …    …
886          0       2  27.0      0      0  13.0000        S  Second    man
887          1       1  19.0      0      0  30.0000        S   First  woman
888          0       3   NaN      1      2  23.4500        S   Third  woman
889          1       1  26.0      0      0  30.0000        C   First    man
890          0       3  32.0      0      0   7.7500        Q   Third    man

     adult_male deck  embark_town alive  alone  female  male
0          True  NaN  Southampton    no  False       0     1
1         False    C    Cherbourg   yes  False       1     0
2         False  NaN  Southampton   yes   True       1     0
3         False    C  Southampton   yes  False       1     0
4          True  NaN  Southampton    no   True       0     1
..          …  …          …   …    …     …  …
886        True  NaN  Southampton    no   True       0     1
887       False    B  Southampton   yes   True       1     0
888       False  NaN  Southampton    no  False       1     0
889        True    C    Cherbourg   yes   True       0     1
890        True  NaN   Queenstown    no   True       0     1

[891 rows x 16 columns]
```