data_wrangling

January 20, 2022

1 Data Wrangling

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
#load dataset
kashti = sns.load_dataset('titanic')
#saving data set into two variable
ks1 = sns.load_dataset('titanic')
#ks2 = kashti
kashti.head(2)
```

[ ]:
```
   survived  pclass     sex  age  sibsp  parch  fare embarked  class    who  \
0         0       3    male   22      1      0     7        S  Third    man
1         1       1  female   38      1      0    71        C  First  woman

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

[ ]:
```
# simple math operation on a series
(kashti['age']+12).head(2)
```

[ ]:
```
0    34
1    50
Name: age, dtype: float64
```

1.1 Dealing with Missing Values

- In a dataset missing values are either ? or NA or NAN or 0 or a blank cell
- Jab data na ho kisi row me kisi bhi ek parameter ka

  Steps: 1. Try recollecting data and check for mistakes. 2. Try to remove missing entries column or remove that entire row 3. Replace the missing values * How ? * Take average value of dat entire data row (column) and substitute null values * Frequency or Mode replacement * Replace based on other functions (Data sampler knows that) * ML algorithms can also be used (like age se salary predict mising) * Leave it like that * Why we deal with the missing values * It is better because no data is lost * Less accurate

```
[ ]: # where exactly missing values are
     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
```

```
[ ]: # use drop.na method
     print(kashti.shape)
     kashti.dropna(subset=["deck"],axis=0, inplace=True)
     # this will remove specifically rows of deck with 0 values
     #inpace = True modifies the frame
```

```
(891, 15)
```

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

```
[ ]: 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
```

[ ]:
```
kashti = kashti.dropna()
kashti.dropna().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           0
embark_town    0
alive          0
alone          0
dtype: int64
```

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

[ ]:
```
(182, 15)
```

[ ]:
```
ks1.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
```

1.2 Replacing missing Values with the average and Mode of that Column

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

[ ]:
```
# replacing NAN with mean of the data (updating as well)
ks1['age'] = ks1['age'].replace(np.nan,mean_age)

ks1['deck'].fillna(ks1['deck'].mode()[0], inplace=True)
ks1['embark_town'].fillna(ks1['embark_town'].mode()[0], inplace=True)
ks1['embarked'].fillna(ks1['embarked'].mode()[0], inplace=True)

#ks1[['deck','embark_town']] = ks1[['age','embark_town']].replace(np.nan,mean)
```

[ ]:
```
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           0
embark_town    0
alive          0
alone          0
dtype: int64
```

1.3 Data Formatting

- Data ko aik common standard par rakhna
- Ensure data is consistent and understandable
  - Easy to gather
  - Easy to work with
    * Faisalabad (FSD)
    * Karachi (KHI)
    * Convert gm to kg or same unit for all.
    * one standard unit

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

```
[ ]: # Convert data type of fixed column(series)     Type Casting
     kashti['survived'] = kashti['survived'].astype('int64')
     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
```

```
[ ]: # convert age into years
     ks1['age'] = ks1['age'] * 365
     #ks1['age'] = pd.set_option('precision', 0)
     ks1.head(3)
```

```
[ ]:    survived  pclass     sex    age  sibsp  parch  fare embarked  class    who  \
     0         0       3    male   8030      1      0     7        S  Third    man
     1         1       1  female  13870      1      0    71        C  First  woman
     2         1       3  female   9490      0      0     8        S  Third  woman
```

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

[ ]:
```
# Renaming    Columns
ks1.rename(columns={"age":"age in Days"},inplace=True)
ks1.head(2)
```

[ ]:
```
   survived  pclass     sex  age in Days  sibsp  parch  fare embarked  class  \
0         0       3    male         8030      1      0     7        S  Third
1         1       1  female        13870      1      0    71        C  First

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

1.4 Data Normalization

- uniform data
- They have same impact
- sea fish vs jar fish
- Also for computational reasons

[ ]:
```
ks4 =ks1[['age in Days','fare']]
ks4.head()
```

[ ]:
```
   age in Days  fare
0         8030     7
1        13870    71
2         9490     8
3        12775    53
4        12775     8
```

1. The above data between fare and age in days is really in wide range. We need to N o r m a l i z e
2. Normalization changes the value to the range of 0 to 1. (both variable will have same influence)

1.4.1 Methods of Normalization

1. Simple feature scaling

- x(new) = x(old) / x(max)

2. Min Max Method

3. Z-score (standard score) -3 to +3

4. Log transformation

```
[ ]: # simple feature scaling
     ks4['fare']= ks4['fare']/ks4['fare'].max()
     ks4['age in Days']= ks4['age in Days']/ks4['age in Days'].max()
     ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/1908861037.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
  ks4['fare']= ks4['fare']/ks4['fare'].max()
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/1908861037.py:3:
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
  ks4['age in Days']= ks4['age in Days']/ks4['age in Days'].max()
```

```
[ ]:   age in Days   fare
     0       3e-01  1e-02
     1       5e-01  1e-01
     2       3e-01  2e-02
     3       4e-01  1e-01
     4       4e-01  2e-02
```

```
[ ]: # 2. Min Max Method
     ks4['fare'] = (ks4['fare']-ks4['fare'].min()) / (ks4['fare'].max() -␣
      ↪ks4['fare'])
     ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/887406347.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
  ks4['fare'] = (ks4['fare']-ks4['fare'].min()) / (ks4['fare'].max() -
ks4['fare'])
```

```
[ ]:   age in Days   fare
     0       3e-01  1e-02
     1       5e-01  2e-01
     2       3e-01  2e-02
```

```
3        4e-01  1e-01
4        4e-01  2e-02
```

[ ]:
```
# z score Method  R A N G E (0   to +3)
ks4['age in Days'] = (ks4['age in Days']-ks4['age in Days'].mean()) /( ks4['age␣
↪in Days'].std() )
ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/4054113253.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
  ks4['age in Days'] = (ks4['age in Days']-ks4['age in Days'].mean()) /(
ks4['age in Days'].std() )
```

[ ]:
```
   age in Days   fare
0       -6e-01  1e-02
1        6e-01  2e-01
2       -3e-01  2e-02
3        4e-01  1e-01
4        4e-01  2e-02
```

[ ]:
```
# 4. log transformation
ks4['fare'] = np.log(ks4['fare'])
ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\arraylike.py:364: RuntimeWarning: divide by zero
encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/2813506387.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
  ks4['fare'] = np.log(ks4['fare'])
```

[ ]:
```
   age in Days  fare
0       -6e-01    -4
1        6e-01    -2
2       -3e-01    -4
3        4e-01    -2
4        4e-01    -4
```

1.5 Binning

1. Grouping of values into small set of values (groups)
2. convert numeric into categories
   1. for example: age (0-10) = bachay 2. age (10-20) = jawan 3. age (30-40) borhay
3. To have better understanding of groups
   1. low vs mid vs high prices

```
[ ]: ks1.sort_values("age in Days")
```

```
[ ]:      survived  pclass     sex  age in Days  sibsp  parch  fare embarked  \
803          1       3    male          153      0      1     9        C
755          1       2    male          245      1      1    14        S
644          1       3  female          274      2      1    19        C
469          1       3  female          274      2      1    19        C
831          1       2    male          303      1      1    19        S
..         …       …     …           …    …     …     …
116          0       3    male        25732      0      0     8        Q
96           0       1    male        25915      0      0    35        C
493          0       1    male        25915      0      0    50        C
851          0       3    male        27010      0      0     8        S
630          1       1    male        29200      0      0    30        S

      class    who  adult_male deck  embark_town alive  alone
803   Third  child       False    C    Cherbourg   yes  False
755  Second  child       False    C  Southampton   yes  False
644   Third  child       False    C    Cherbourg   yes  False
469   Third  child       False    C    Cherbourg   yes  False
831  Second  child       False    C  Southampton   yes  False
..      …      …          …    …          …     …     …
116   Third    man        True    C   Queenstown    no   True
96    First    man        True    A    Cherbourg    no   True
493   First    man        True    C    Cherbourg    no   True
851   Third    man        True    C  Southampton    no   True
630   First    man        True    A  Southampton   yes   True

[891 rows x 15 columns]
```

```
[ ]: # bins = np.linspace(min(ks1['age in Days']), max(ks1['age in Days']) , 29200)
     # age_groups = ["Bachay","Jawaan","Boorhay"]
     # ks1['age in Days']=pd.cut(ks1['age in Days'],bins, labels=age_groups,␣
      ↪include_lowest=True)
     # ks1['age in Days']
```

```
[ ]: kashti["age_bin"] = pd.cut(kashti["age"],bins=[0,2,17,65,99],
        labels=['Toddler/baby','Child','Adult','Elderly'])
```

```
[ ]: kashti
[ ]:      survived  pclass     sex  age  sibsp  parch  fare embarked  class    who  \
     1           1       1  female   38      1      0    71        C  First  woman
     3           1       1  female   35      1      0    53        S  First  woman
     6           0       1    male   54      0      0    52        S  First    man
     10          1       3  female    4      1      1    17        S  Third  child
     11          1       1  female   58      0      0    27        S  First  woman
     ..        …       …      …  …      …     …     …       …     …     …
     871         1       1  female   47      1      1    53        S  First  woman
     872         0       1    male   33      0      0     5        S  First    man
     879         1       1  female   56      0      1    83        C  First  woman
     887         1       1  female   19      0      0    30        S  First  woman
     889         1       1    male   26      0      0    30        C  First    man

          adult_male deck  embark_town alive  alone age_bin
     1         False    C    Cherbourg   yes  False   Adult
     3         False    C  Southampton   yes  False   Adult
     6          True    E  Southampton    no   True   Adult
     10        False    G  Southampton   yes  False   Child
     11        False    C  Southampton   yes   True   Adult
     ..          …   …           …     …     …       …
     871       False    D  Southampton   yes  False   Adult
     872        True    B  Southampton    no   True   Adult
     879       False    C    Cherbourg   yes  False   Adult
     887       False    B  Southampton   yes   True   Adult
     889        True    C    Cherbourg   yes   True   Adult

     [182 rows x 16 columns]
```

1.6 Dummies

```
[ ]: ks1
```

```
[ ]:      survived  pclass     sex  age in Days  sibsp  parch  fare embarked  \
     0           0       3    male         8030      1      0     7        S
     1           1       1  female        13870      1      0    71        C
     2           1       3  female         9490      0      0     8        S
     3           1       1  female        12775      1      0    53        S
     4           0       3    male        12775      0      0     8        S
     ..        …       …      …           …     …     …     …       …
     886         0       2    male         9855      0      0    13        S
     887         1       1  female         6935      0      0    30        S
     888         0       3  female        10840      1      2    23        S
     889         1       1    male         9490      0      0    30        C
     890         0       3    male        11680      0      0     8        Q

           class    who  adult_male deck  embark_town alive  alone
     0     Third    man        True    C  Southampton    no  False
```

```
1     First  woman       False    C    Cherbourg   yes  False
2     Third  woman       False    C  Southampton   yes   True
3     First  woman       False    C  Southampton   yes  False
4     Third    man        True    C  Southampton    no   True
..      …      …       …   …           …    …     …
886  Second    man        True    C  Southampton    no   True
887   First  woman       False    B  Southampton   yes   True
888   Third  woman       False    C  Southampton    no  False
889   First    man        True    C    Cherbourg   yes   True
890   Third    man        True    C   Queenstown    no   True

[891 rows x 15 columns]
```

[ ]:
```
# converting categories to dummy values
pd.get_dummies(ks1['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]
```

[ ]:
```
ks1 =pd.concat([ks1, pd.get_dummies(ks1['sex'])], axis=1)
ks1 =ks1.drop("sex", axis=1)      #####
```

[ ]:
```
ks1
```

[ ]:
```
     survived  pclass  age in Days  sibsp  parch  fare embarked   class  \
0           0       3         8030      1      0     7        S   Third
1           1       1        13870      1      0    71        C   First
2           1       3         9490      0      0     8        S   Third
3           1       1        12775      1      0    53        S   First
4           0       3        12775      0      0     8        S   Third
..        …     …           …    …    …    …       …    …
886         0       2         9855      0      0    13        S  Second
887         1       1         6935      0      0    30        S   First
888         0       3        10840      1      2    23        S   Third
889         1       1         9490      0      0    30        C   First
```

```
890        0       3        11680      0      0     8        Q   Third

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

[891 rows x 16 columns]
```

```
[ ]:
```