

# eda

March 16, 2022

## 0.1 Exploratory Data Analysis

This will show us how can we do EDA using python

### 0.1.1 Three important steps to keep in mind are:

- 1- Understand the Data.
- 2- clean the data.
- 3- Find a relationship between data.

```
[ ]: # import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

load seaborn library dataset

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

```
[ ]:      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
..          ...      ...    ...   ...   ...   ...   ...   ...
886          0         2   male  27.0     0     0  13.0000         S  Second
887          1         1 female  19.0     0     0  30.0000         S   First
888          0         3 female   NaN     1     2  23.4500         S   Third
889          1         1   male  26.0     0     0  30.0000         C   First
890          0         3   male  32.0     0     0   7.7500         Q   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
...	...	...	...	...	...	...
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True
890	man	True	NaN	Queenstown	no	True

[891 rows x 15 columns]

to save titanic data set as csv file

```
[ ]: kashti.to_csv('kashti.csv')
```

### 0.1.2 1- Understanding the data

to get full information of kashti dataset

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column          Non-Null Count  Dtype
---  -
0   survived        891 non-null    int64
1   pclass          891 non-null    int64
2   sex             891 non-null    object
3   age            714 non-null    float64
4   sibsp          891 non-null    int64
5   parch          891 non-null    int64
6   fare           891 non-null    float64
7   embarked       889 non-null    object
8   class          891 non-null    category
9   who            891 non-null    object
10  adult_male     891 non-null    bool
11  deck          203 non-null    category
12  embark_town    889 non-null    object
13  alive          891 non-null    object
14  alone          891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
[ ]: ks = kashti
```

to print header of data

```
[ ]: ks.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
```

to print shape of dataset

```
[ ]: ks.shape
```

```
[ ]: (891, 15)
```

to print tail of dataset

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

```
[ ]:      survived  pclass      sex   age  sibsp  parch      fare embarked  class \
886          0        2    male  27.0     0     0   13.00          S  Second
887          1        1  female  19.0     0     0   30.00          S  First
888          0        3  female  NaN     1     2   23.45          S  Third
889          1        1    male  26.0     0     0   30.00          C  First
890          0        3    male  32.0     0     0    7.75          Q  Third

      who  adult_male deck  embark_town alive  alone
886  man          True  NaN  Southampton    no  True
887  woman          False   B  Southampton   yes  True
888  woman          False  NaN  Southampton    no  False
889  man          True   C   Cherbourg   yes  True
890  man          True  NaN  Queenstown    no  True
```

to get information of numeric variable

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

```
[ ]:      survived      pclass      age      sibsp      parch      fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean    0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std     0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min     0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%     0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%     0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
```

75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

to print unique values of a column in dataset

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

```
[ ]: survived      2
     pclass        3
     sex           2
     age          88
     sibsp         7
     parch         7
     fare         248
     embarked      3
     class         3
     who           3
     adult_male    2
     deck          7
     embark_town   3
     alive         2
     alone         2
     dtype: int64
```

to print all column names of dataset

```
[ ]: ks.columns
```

```
[ ]: Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
           'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
           'alive', 'alone'],
           dtype='object')
```

```
[ ]: ks['sex'].unique()    #nunique print number of unique values and unique print,
    ↪ the values
```

```
[ ]: array(['male', 'female'], dtype=object)
```

```
[ ]: ks['embarked'].unique()
```

```
[ ]: array(['S', 'C', 'Q', nan], dtype=object)
```

### 0.1.3 Assignment:

How to print unique values of all column in one line of code.

```
[ ]: ks_series=ks.columns
     for i in ks_series:
         print("The unique value of:", i)
```

```
print(ks[i].unique())
```

The unique value of: survived

[0 1]

The unique value of: pclass

[3 1 2]

The unique value of: sex

['male' 'female']

The unique value of: age

```
[22.  38.  26.  35.   nan 54.   2.  27.  14.   4.  58.  20.
 39.  55.  31.  34.  15.  28.   8.  19.  40.  66.  42.  21.
 18.   3.   7.  49.  29.  65. 28.5   5.  11.  45.  17.  32.
 16.  25.  0.83 30.  33.  23.  24.  46.  59.  71.  37.  47.
 14.5 70.5 32.5 12.   9.  36.5 51.  55.5 40.5 44.   1.  61.
 56.  50.  36.  45.5 20.5 62.  41.  52.  63.  23.5  0.92 43.
 60.  10.  64.  13.  48.   0.75 53.  57.  80.  70.  24.5   6.
  0.67 30.5  0.42 34.5 74. ]
```

The unique value of: sibsp

[1 0 3 4 2 5 8]

The unique value of: parch

[0 1 2 5 3 4 6]

The unique value of: fare

```
[ 7.25  71.2833  7.925  53.1    8.05   8.4583 51.8625 21.075
 11.1333 30.0708 16.7    26.55  31.275  7.8542 16.    29.125
 13.    18.    7.225  26.    8.0292 35.5    31.3875 263.
 7.8792 7.8958 27.7208 146.5208 7.75    10.5    82.1708 52.
 7.2292 11.2417 9.475   21.    41.5792 15.5    21.6792 17.8
 39.6875 7.8    76.7292 61.9792 27.75   46.9    80.    83.475
 27.9    15.2458 8.1583  8.6625 73.5    14.4542 56.4958 7.65
 29.    12.475  9.    9.5    7.7875 47.1    15.85  34.375
 61.175 20.575 34.6542 63.3583 23.    77.2875 8.6542 7.775
 24.15  9.825  14.4583 247.5208 7.1417 22.3583 6.975  7.05
 14.5    15.0458 26.2833 9.2167 79.2    6.75    11.5    36.75
 7.7958 12.525 66.6    7.3125 61.3792 7.7333 69.55  16.1
 15.75  20.525 55.    25.925 33.5    30.6958 25.4667 28.7125
 0.    15.05  39.    22.025 50.    8.4042 6.4958 10.4625
 18.7875 31.    113.275 27.    76.2917 90.    9.35  13.5
 7.55  26.25  12.275  7.125 52.5542 20.2125 86.5  512.3292
 79.65 153.4625 135.6333 19.5  29.7    77.9583 20.25  78.85
 91.0792 12.875 8.85  151.55 30.5    23.25  12.35 110.8833
 108.9  24.    56.9292 83.1583 262.375 14.    164.8667 134.5
 6.2375 57.9792 28.5  133.65 15.9    9.225  35.    75.25
 69.3  55.4417 211.5  4.0125 227.525 15.7417 7.7292 12.
 120.  12.65  18.75  6.8583 32.5    7.875  14.4  55.9
 8.1125 81.8583 19.2583 19.9667 89.1042 38.5    7.725 13.7917
 9.8375 7.0458 7.5208 12.2875 9.5875 49.5042 78.2667 15.1
 7.6292 22.525 26.2875 59.4    7.4958 34.0208 93.5  221.7792
 106.425 49.5  71.    13.8625 7.8292 39.6    17.4  51.4792
```

```

26.3875 30.      40.125   8.7125 15.      33.      42.4    15.55
65.      32.3208  7.0542  8.4333 25.5875  9.8417  8.1375 10.1708
211.3375 57.      13.4167  7.7417  9.4833  7.7375  8.3625 23.45
25.9292  8.6833  8.5167  7.8875 37.0042  6.45    6.95    8.3
6.4375  39.4    14.1083 13.8583 50.4958  5.      9.8458 10.5167]
The unique value of: embarked
['S' 'C' 'Q' nan]
The unique value of: class
['Third', 'First', 'Second']
Categories (3, object): ['First', 'Second', 'Third']
The unique value of: who
['man' 'woman' 'child']
The unique value of: adult_male
[ True False]
The unique value of: deck
[NaN, 'C', 'E', 'G', 'D', 'A', 'B', 'F']
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
The unique value of: embark_town
['Southampton' 'Cherbourg' 'Queenstown' nan]
The unique value of: alive
['no' 'yes']
The unique value of: alone
[False  True]

```

#### 0.1.4 2- Cleaning and filtering the data

to find missing values in dataset

```
[ ]: ks.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

```

to remove the column with most missing values (cleaning)

```
[ ]: ks_clean = ks.drop(['deck'], axis=1)
ks_clean.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  embark_town  alive  alone
0   man          True  Southampton    no  False
1 woman         False   Cherbourg   yes  False
2 woman         False  Southampton   yes   True
3 woman         False  Southampton   yes  False
4   man          True  Southampton    no   True
```

```
[ ]: ks_clean.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
embark_town      2
alive            0
alone            0
dtype: int64
```

```
[ ]: ks_clean.shape
```

```
[ ]: (891, 14)
```

to drop all NaN from dataframe

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

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

```

```
[ ]: ks_clean.shape
```

```
[ ]: (712, 14)
```

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

```

[ ]:
      survived      pclass      age      sibsp      parch      fare
count  712.000000  712.000000  712.000000  712.000000  712.000000  712.000000
mean    0.404494    2.240169   29.642093    0.514045    0.432584   34.567251
std     0.491139    0.836854   14.492933    0.930692    0.854181   52.938648
min     0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%     0.000000    1.000000   20.000000    0.000000    0.000000    8.050000
50%     0.000000    2.000000   28.000000    0.000000    0.000000   15.645850
75%     1.000000    3.000000   38.000000    1.000000    1.000000   33.000000
max     1.000000    3.000000   80.000000    5.000000    6.000000  512.329200

```

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

```

[ ]:
      survived      pclass      age      sibsp      parch      fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean    0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std     0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min     0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%     0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%     0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%     1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max     1.000000    3.000000   80.000000    8.000000    6.000000  512.329200

```

```
[ ]: ks_clean.columns
```

```

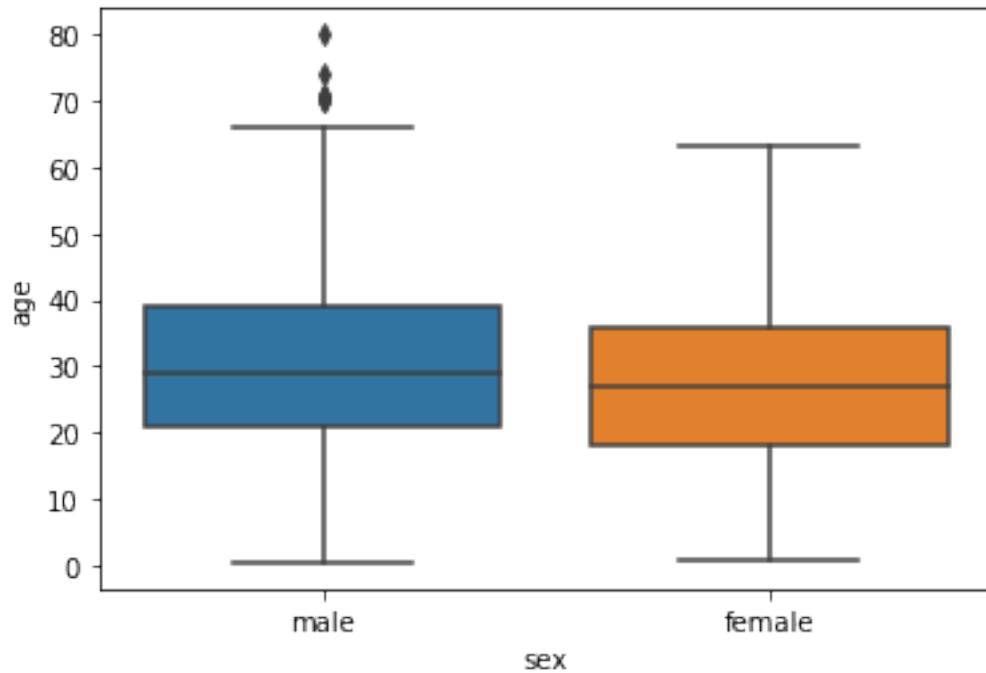
[ ]: Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
          'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
          'alone'],
          dtype='object')

```



```
[ ]: sns.boxplot(x = 'sex', y = 'age', data=ks_clean)
```

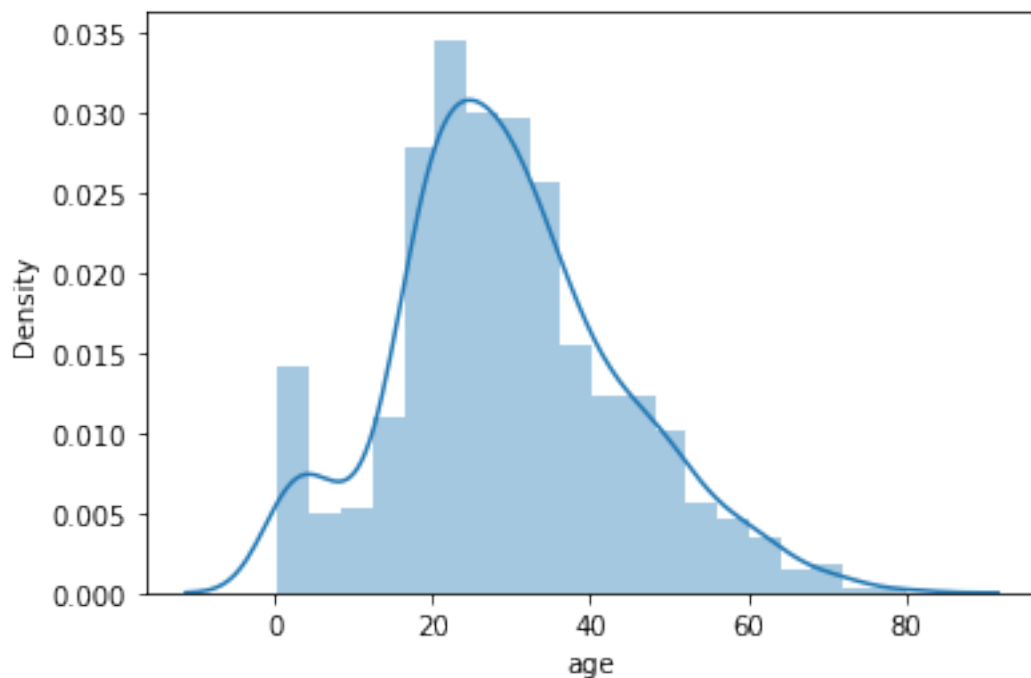
```
[ ]: <AxesSubplot:xlabel='sex', ylabel='age'>
```



```
[ ]: sns.distplot(ks_clean['age']) # the below graph is called bell curve, □  
    ↪ histogram, normality check or distance plt.
```

C:\Users\Faisal Hayat\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)

```
[ ]: <AxesSubplot:xlabel='age', ylabel='Density'>
```



Remarks: To make bell shape curve we apply diff method one method is to remove outlier

Dispersion and mean has to be seen

```
[ ]: # To remove outliers
ks_clean['age'].mean()
```

```
[ ]: 29.64209269662921
```

```
[ ]: ks_clean = ks_clean[ks_clean['age'] < 65]
ks_clean
```

```
[ ]:      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
..      ...      ...      ...      ...      ...      ...
885          0      3 female  39.0     0     5  29.1250         Q   Third
886          0      2  male  27.0     0     0  13.0000         S  Second
887          1      1 female  19.0     0     0  30.0000         S   First
889          1      1  male  26.0     0     0  30.0000         C   First
890          0      3  male  32.0     0     0   7.7500         Q   Third
```

	who	adult_male	embark_town	alive	alone
0	man	True	Southampton	no	False
1	woman	False	Cherbourg	yes	False
2	woman	False	Southampton	yes	True
3	woman	False	Southampton	yes	False
4	man	True	Southampton	no	True
..	...	...	...	...	...
885	woman	False	Queenstown	no	False
886	man	True	Southampton	no	True
887	woman	False	Southampton	yes	True
889	man	True	Cherbourg	yes	True
890	man	True	Queenstown	no	True

[701 rows x 14 columns]

```
[ ]: ks_clean.shape
```

```
[ ]: (701, 14)
```

```
[ ]: ks_clean['age'].mean()
```

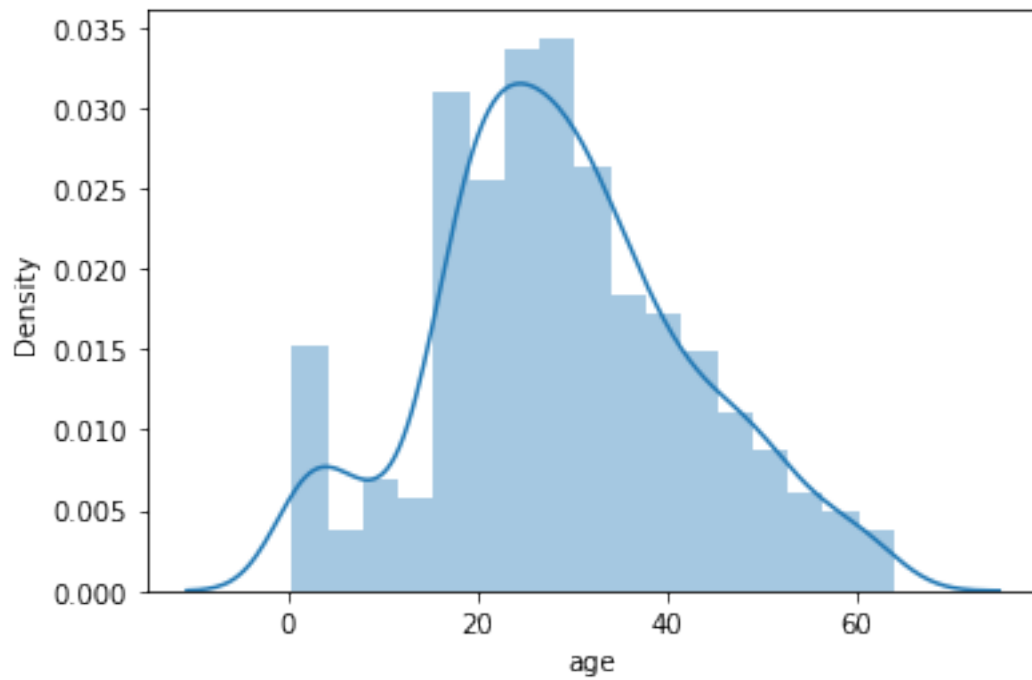
```
[ ]: 29.01236804564907
```

```
[ ]: sns.distplot(ks_clean['age'])
```

C:\Users\Faisal Hayat\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

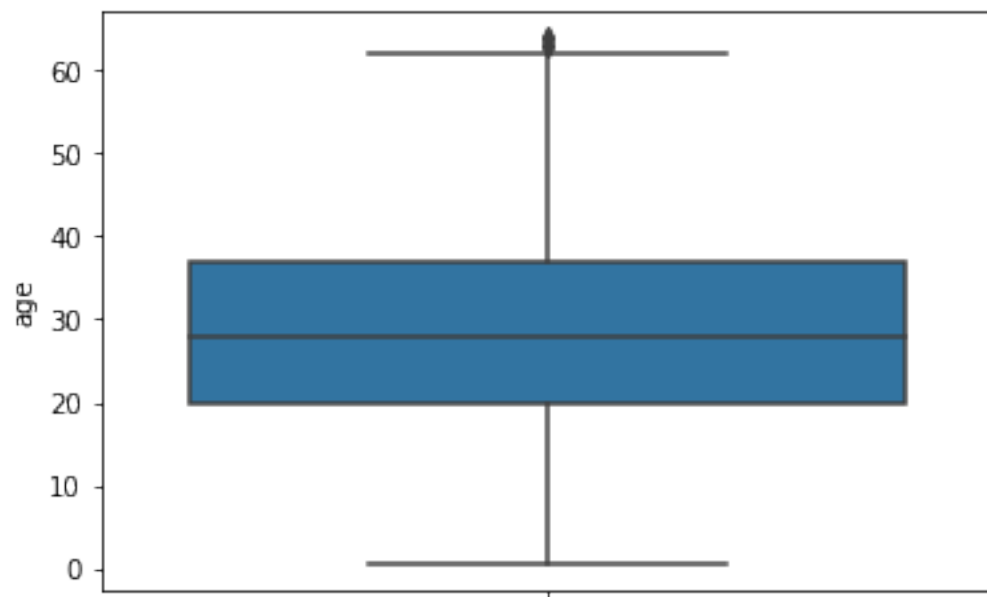
warnings.warn(msg, FutureWarning)

```
[ ]: <AxesSubplot:xlabel='age', ylabel='Density'>
```



```
[ ]: sns.boxplot(y='age', data=ks_clean)
```

```
[ ]: <AxesSubplot:ylabel='age'>
```



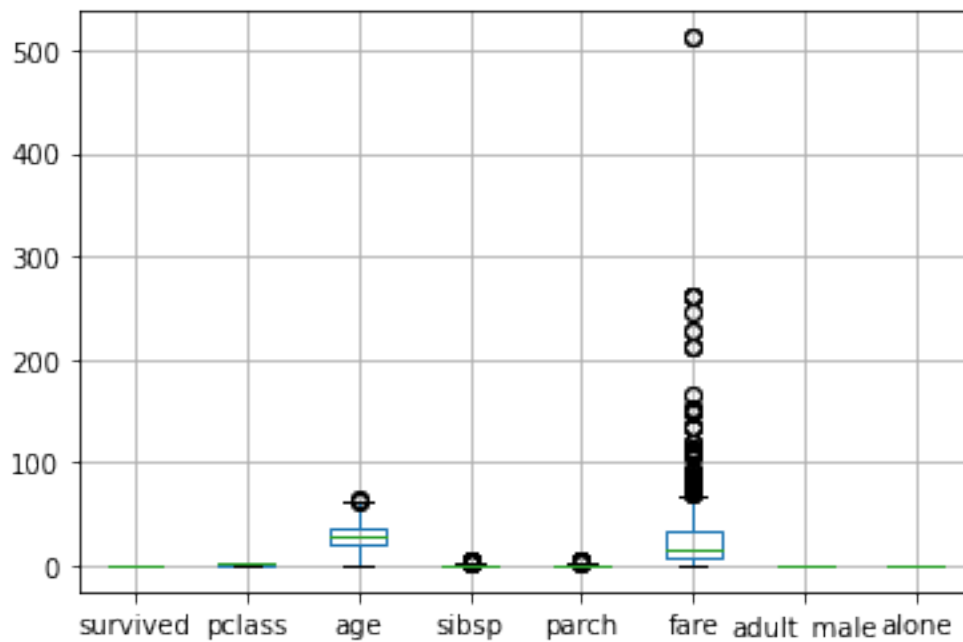
```
[ ]: ks_clean.head()
```

```
[ ]:   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  embark_town  alive  alone
0    man          True  Southampton    no  False
1  woman         False   Cherbourg   yes  False
2  woman         False  Southampton   yes   True
3  woman         False  Southampton   yes  False
4    man          True  Southampton    no   True
```

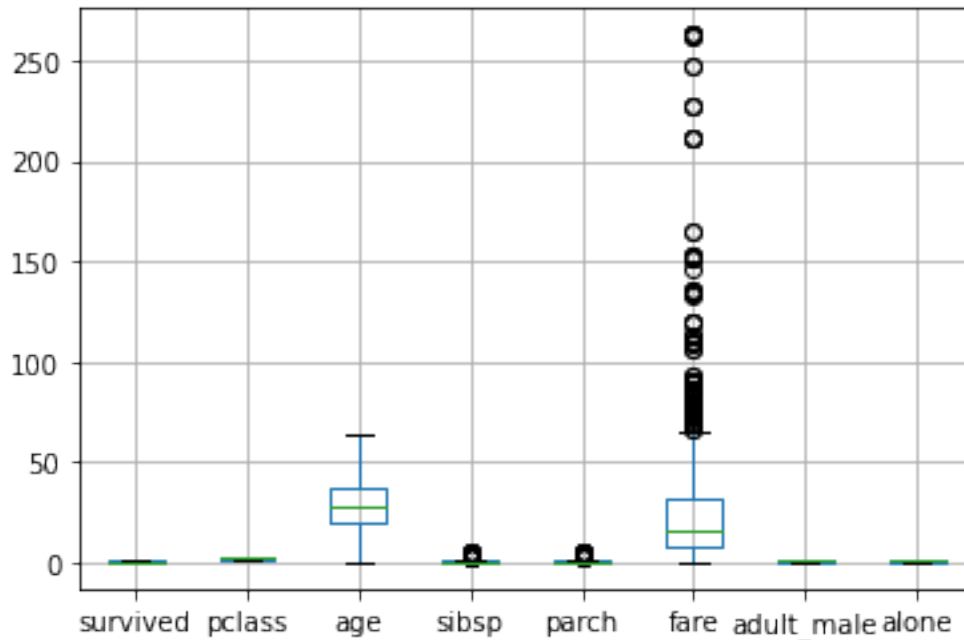
```
[ ]: ks_clean.boxplot()
```

```
[ ]: <AxesSubplot:>
```



```
[ ]: ks_clean = ks_clean[ks_clean['fare']<300]
ks_clean.boxplot()
```

```
[ ]: <AxesSubplot:>
```

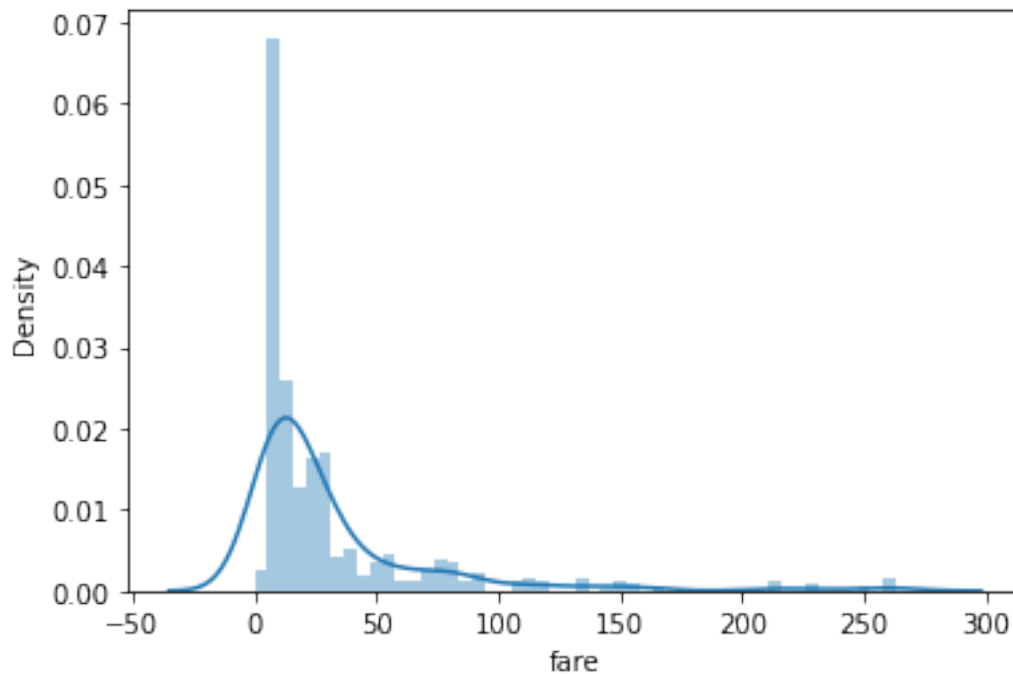


```
[ ]: sns.distplot(ks_clean['fare'])
```

C:\Users\Faisal Hayat\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

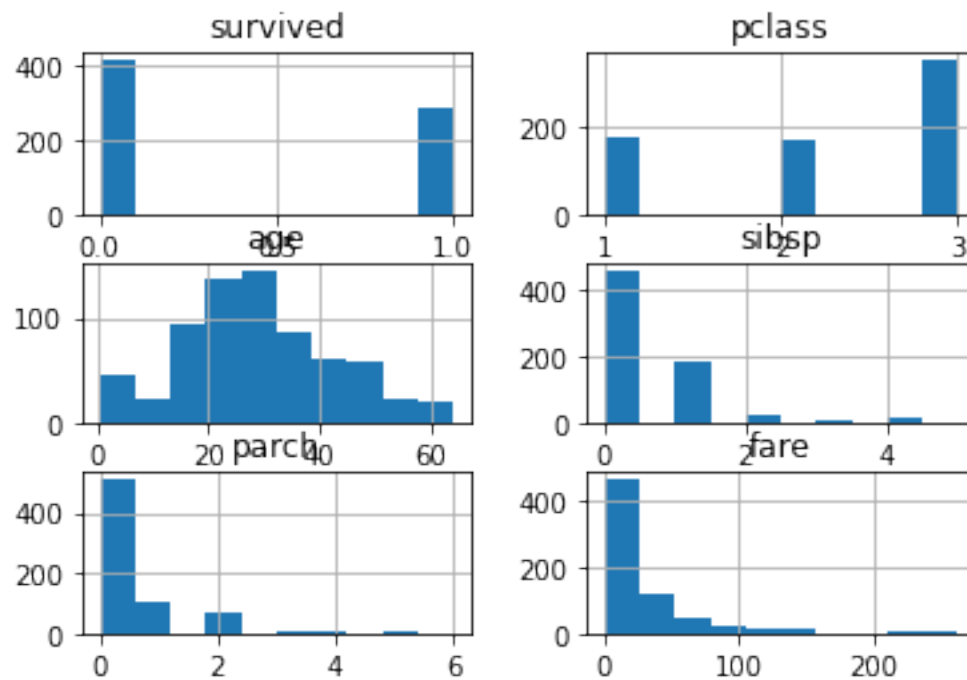
warnings.warn(msg, FutureWarning)

```
[ ]: <AxesSubplot:xlabel='fare', ylabel='Density'>
```



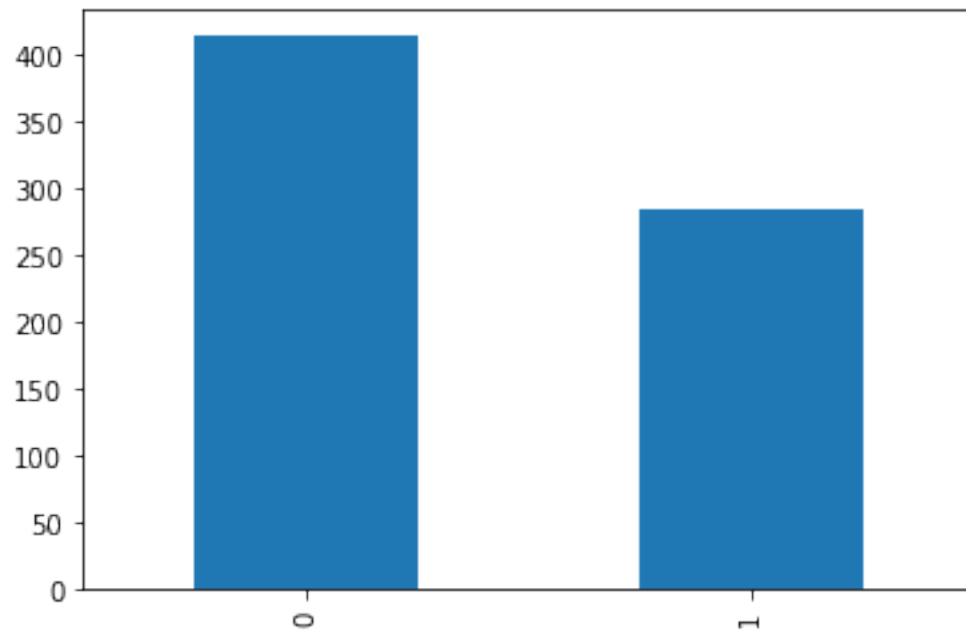
```
[ ]: # to make histogram of whole dataset  
ks_clean.hist()
```

```
[ ]: array([[<AxesSubplot:title={'center': 'survived'}>,  
          <AxesSubplot:title={'center': 'pclass'}>],  
          [<AxesSubplot:title={'center': 'age'}>,  
          <AxesSubplot:title={'center': 'sibsp'}>],  
          [<AxesSubplot:title={'center': 'parch'}>,  
          <AxesSubplot:title={'center': 'fare'}>]], dtype=object)
```



```
[ ]: pd.value_counts(ks_clean['survived']).plot.bar()
```

```
[ ]: <AxesSubplot:>
```





```
[ ]: ks_clean.groupby(['sex']).mean()
```

```
[ ]:      survived    pclass      age    sibsp    parch      fare \
sex
female  0.751938  2.077519  27.717054  0.647287  0.717054  45.530120
male    0.204545  2.356818  29.728795  0.450000  0.272727  25.023095

      adult_male    alone
sex
female    0.000000  0.375969
male      0.909091  0.668182
```

```
[ ]: ks_clean.groupby(['sex', 'class']).mean()
```

```
[ ]:      survived    pclass      age    sibsp    parch      fare \
sex    class
female First  0.963415      1.0  34.231707  0.560976  0.512195  103.696393
       Second 0.918919      2.0  28.722973  0.500000  0.621622   21.951070
       Third  0.460784      3.0  21.750000  0.823529  0.950980   15.875369
male   First  0.397849      1.0  39.531398  0.397849  0.333333   63.301881
       Second 0.154639      2.0  29.972474  0.381443  0.247423   21.331959
       Third  0.152000      3.0  25.987680  0.496000  0.260000   12.215548

      adult_male    alone
sex    class
female First    0.000000  0.353659
       Second   0.000000  0.405405
       Third    0.000000  0.372549
male   First    0.967742  0.526882
       Second   0.907216  0.628866
       Third    0.888000  0.736000
```

```
[ ]: ks_clean
```

```
[ ]:      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
..      ...      ...      ...      ...      ...      ...      ...      ...
885      0         3  female    39.0      0      5   29.1250      Q    Third
886      0         2    male    27.0      0      0   13.0000      S    Second
887      1         1  female    19.0      0      0   30.0000      S    First
889      1         1    male    26.0      0      0   30.0000      C    First
890      0         3    male    32.0      0      0    7.7500      Q    Third
```

	who	adult_male	embark_town	alive	alone
0	man	True	Southampton	no	False
1	woman	False	Cherbourg	yes	False
2	woman	False	Southampton	yes	True
3	woman	False	Southampton	yes	False
4	man	True	Southampton	no	True
..	...	...	...	...	...
885	woman	False	Queenstown	no	False
886	man	True	Southampton	no	True
887	woman	False	Southampton	yes	True
889	man	True	Cherbourg	yes	True
890	man	True	Queenstown	no	True

[698 rows x 14 columns]

```
[ ]: ks
```

```
[ ]:
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
..      ...      ...   ...   ...   ...   ...   ...   ...
886       0        2   male  27.0    0     0  13.0000         S  Second
887       1        1  female  19.0    0     0  30.0000         S   First
888       0        3  female   NaN    1     2  23.4500         S   Third
889       1        1   male  26.0    0     0  30.0000         C   First
890       0        3   male  32.0    0     0   7.7500         Q   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
..	...	...	...	...	...	...
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True
890	man	True	NaN	Queenstown	no	True

[891 rows x 15 columns]

### 0.1.5 3- Relationship

```
[ ]: # to find correlation of data set, below function will give us correlation
      ↪matrix
ks_clean.corr()
```

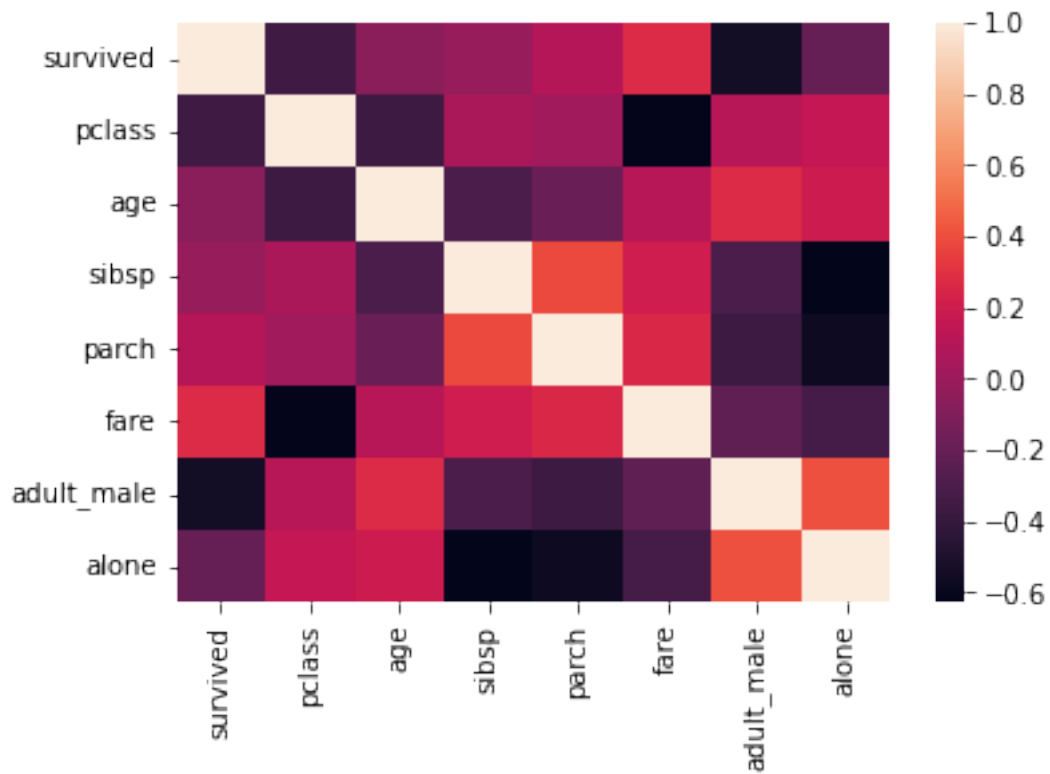
```
[ ]:
survived    pclass    age    sibsp    parch    fare \
survived    1.000000 -0.361495 -0.063460 -0.017159 0.094668 0.273631
pclass      -0.361495 1.000000 -0.364680 0.059720 0.023379 -0.618783
age         -0.063460 -0.364680 1.000000 -0.306870 -0.186806 0.107324
sibsp       -0.017159 0.059720 -0.306870 1.000000 0.381779 0.197847
parch       0.094668 0.023379 -0.186806 0.381779 1.000000 0.258839
fare        0.273631 -0.618783 0.107324 0.197847 0.258839 1.000000
adult_male  -0.552755 0.106530 0.268049 -0.309813 -0.366564 -0.228663
alone       -0.200286 0.155759 0.185661 -0.630175 -0.573696 -0.332415

      adult_male    alone
survived    -0.552755 -0.200286
pclass       0.106530 0.155759
age          0.268049 0.185661
sibsp        -0.309813 -0.630175
parch        -0.366564 -0.573696
fare         -0.228663 -0.332415
adult_male   1.000000 0.402231
alone        0.402231 1.000000
```

```
[ ]: cor_var = ks_clean.corr()
```

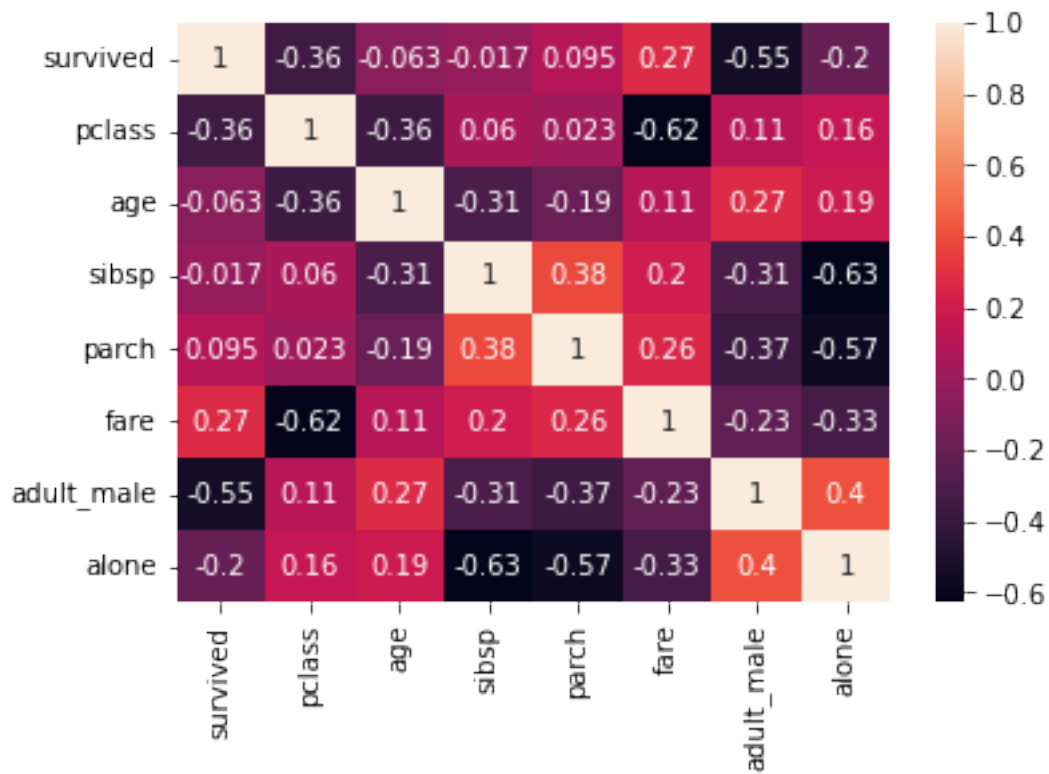
```
[ ]: # drawing heatmap
sns.heatmap(cor_var)
```

```
[ ]: <AxesSubplot:>
```



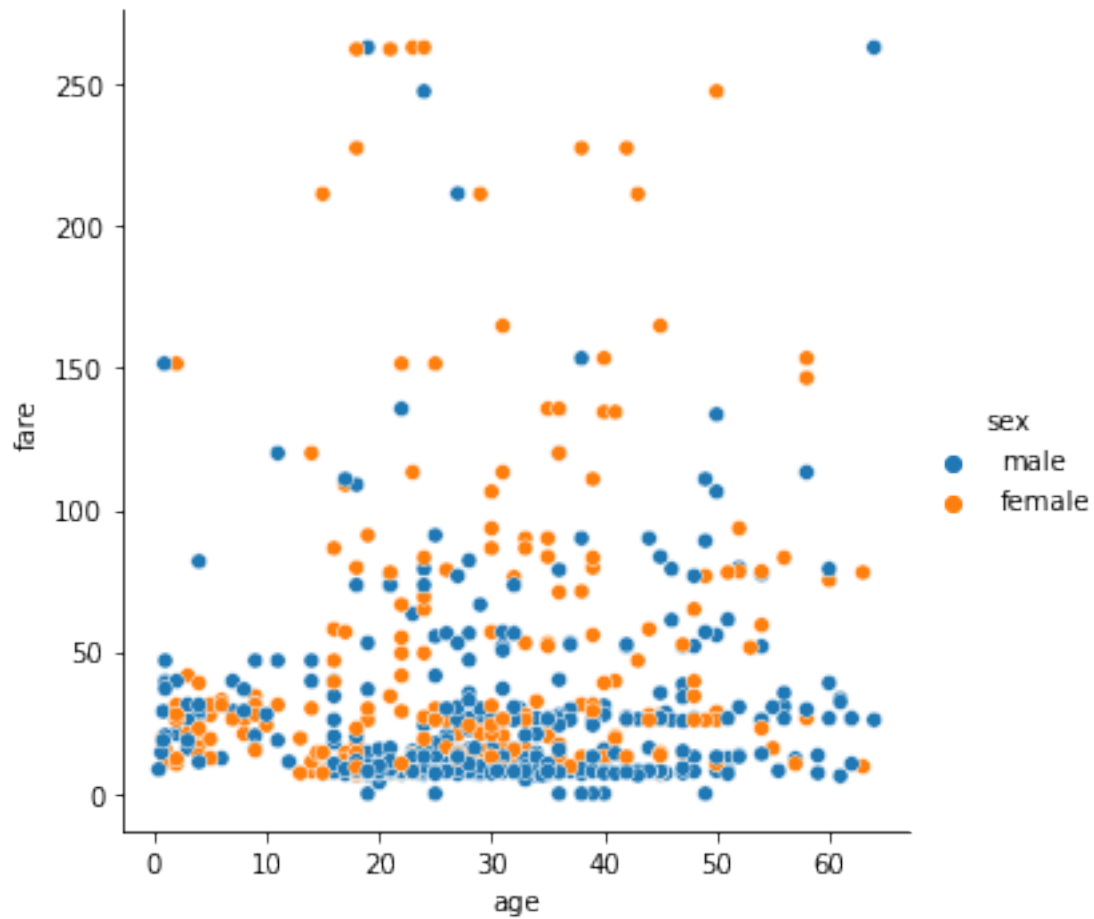
```
[ ]: sns.heatmap(cor_var, annot=True)
```

```
[ ]: <AxesSubplot:>
```



```
[ ]: # correlation is actually for numerical variable
sns.relplot(x='age', y='fare', hue='sex', data=ks_clean)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x286d5f76da0>
```



```
[ ]: sns.catplot(x='sex', y='fare', hue='sex', data=ks_clean, kind='bar') # kind ↵
    ↪ can be box
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x286d5ea0e80>
```

