Manaranjan Pradhan
manaranjan@enablecloud.com
*This notebook is given as part of **Data Science for everyone** workshop.*
(Forwarding this document to others is strictly prohibited.)

Working with Pandas - DataFrame ¶

Importing pandas and numpy library

In [1]:

```
import pandas as pd
import numpy as np
```

Read the dataset, which is in csv format

In [2]:

```
titanic_data = pd.read_csv("titanic.csv")
```

Print the first few rows

In [3]:

```
titanic_data.head()
```

Out[3]:

	row.names	pclass	survived	name	age	embarked	home.dest
0	1	1st	1	Allen, Miss Elisabeth Walton	29.0000	Southampton	St Louis, MO
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chesterville, ON
2	3	1st	0	Allison, Mr Hudson Joshua Creighton	30.0000	Southampton	Montreal, PQ / Chesterville, ON
3	4	1st	0	Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)	25.0000	Southampton	Montreal, PQ / Chesterville, ON
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chesterville, ON

Check dataset dimensions.. how many row and columns?

In [4]:

```
titanic_data.shape
```

Out[4]:

```
(1313, 11)
```

List column names

In [5]:

```
titanic_data.columns
```

Out[5]:

```
Index(['row.names', 'pclass', 'survived', 'name', 'age', 'embarked',
       'home.dest', 'room', 'ticket', 'boat', 'sex'],
      dtype='object')
```

Print column types

In [6]:

```
titanic_data.dtypes
```

Out[6]:

```
row.names      int64
pclass        object
survived       int64
name          object
age          float64
embarked      object
home.dest     object
room          object
ticket        object
boat          object
sex           object
dtype: object
```

Some more information with info() command

How many total entries? What are the columns and their types. Each column has how many not-null values?

In [7]:

```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1313 entries, 0 to 1312
Data columns (total 11 columns):
row.names    1313 non-null int64
pclass       1313 non-null object
survived     1313 non-null int64
name         1313 non-null object
age          633 non-null float64
embarked     821 non-null object
home.dest    754 non-null object
room         77 non-null object
ticket       69 non-null object
boat         347 non-null object
sex          1313 non-null object
dtypes: float64(1), int64(2), object(8)
memory usage: 123.1+ KB
```

Select Specific columns and print

In [8]:

```
titanic_data['survived'][0:10]
## or titanic_data.survived
```

Out[8]:

```
0    1
1    0
2    0
3    0
4    1
5    1
6    1
7    0
8    1
9    0
Name: survived, dtype: int64
```

In [9]:

```
titanic_data.survived[0:10]
```

Out[9]:

```
0    1
1    0
2    0
3    0
4    1
5    1
6    1
7    0
8    1
9    0
Name: survived, dtype: int64
```

In [10]:

```
## Selecting multiple columns

titanic_data[['survived','age']][0:10]
```

Out[10]:

	survived	age
0	1	29.0000
1	0	2.0000
2	0	30.0000
3	0	25.0000
4	1	0.9167
5	1	47.0000
6	1	63.0000
7	0	39.0000
8	1	58.0000
9	0	71.0000

How many people survived and what is the percentage?

In [11]:

```
titanic_data['survived'].value_counts()
```

Out[11]:

```
0    864
1    449
dtype: int64
```

In [12]:

```
titanic_data['survived'].value_counts(normalize=True) * 100
```

Out[12]:

```
0    65.803503
1    34.196497
dtype: float64
```

Get quick statistics of variables

In [13]:

```
titanic_data.describe()
```

Out[13]:

	row.names	survived	age
count	1313.000000	1313.000000	633.000000
mean	657.000000	0.341965	31.194181
std	379.174762	0.474549	14.747525
min	1.000000	0.000000	0.166700
25%	329.000000	0.000000	21.000000
50%	657.000000	0.000000	30.000000
75%	985.000000	1.000000	41.000000
max	1313.000000	1.000000	71.000000

pd.corsstab() function is used for categorical variables.

In [14]:

```
pd.crosstab( titanic_data.sex, titanic_data.survived )
```

Out[14]:

survived	0	1
sex		
female	156	307
male	708	142

In [15]:

```
pd.crosstab( titanic_data.pclass, titanic_data.survived )
```

Out[15]:

survived	0	1
pclass		
1st	129	193
2nd	161	119
3rd	574	137

In [16]:

```
pd.crosstab( titanic_data.pclass, titanic_data.embarked )
```

Out[16]:

embarked	**Cherbourg**	**Queenstown**	**Southampton**
pclass			
1st	142	3	167
2nd	28	7	237
3rd	33	35	169

In [17]:

```
pd.crosstab( titanic_data.survived, titanic_data.embarked )
```

Out[17]:

embarked	**Cherbourg**	**Queenstown**	**Southampton**
survived			
0	84	31	344
1	119	14	229

Filtering records based on a condition

How many children survived who are less than 5 years old?

In [18]:

```
below_5_years = titanic_data[ titanic_data.age <= 5 ]
```

In [19]:

```
below_5_years[0:5]
```

Out[19]:

	row.names	pclass	survived	name	age	embarked	home.des
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chestervil ON
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chestervil ON
86	87	1st	1	Dodge, Master Washington	4.0000	Southampton	San Francisco CA
338	339	2nd	1	Becker, Miss Marion Louise	4.0000	Southampton	Guntur, India / Benton Harbour,
339	340	2nd	1	Becker, Master Richard F.	1.0000	Southampton	Guntur, India / Benton Harbour,

In [20]:

```
len( titanic_data[ titanic_data.age <= 5 ] )
```

Out[20]:

```
29
```

In [21]:

```
titanic_data[ titanic_data.age <= 5 ]["survived"].value_counts()
```

Out[21]:

```
1    24
0     5
dtype: int64
```

In [22]:

```
titanic_data[ titanic_data.age <= 5 ]["survived"].value_counts( normalize = True )
```

Out[22]:

```
1    0.827586
0    0.172414
dtype: float64
```

In [23]:

```
titanic_data.columns
```

Out[23]:

```
Index(['row.names', 'pclass', 'survived', 'name', 'age', 'embarked',
       'home.dest', 'room', 'ticket', 'boat', 'sex'],
      dtype='object')
```

Get unique values for a column

How many embark points were there? dataframe.unique() lists unique values of the column

In [24]:

```
titanic_data.embarked.unique()
```

Out[24]:

```
array(['Southampton', 'Cherbourg', nan, 'Queenstown'], dtype=object)
```

Working with NA values

Count and drop NA Values

In [25]:

```
titanic_data.embarked.unique()
```

Out[25]:

```
array(['Southampton', 'Cherbourg', nan, 'Queenstown'], dtype=object)
```

In [26]:

```
len( titanic_data )
```

Out[26]:

```
1313
```

In [27]:

```
titanic_data.embarked.dropna().unique()
```

Out[27]:

```
array(['Southampton', 'Cherbourg', 'Queenstown'], dtype=object)
```

In [28]:

```
len( titanic_data.embarked.dropna().unique() )
```

Out[28]:

```
3
```

In [29]:

```
len( titanic_data[ titanic_data.embarked.notnull() ] )
```

Out[29]:

```
821
```

In [30]:

```
len( titanic_data[ titanic_data.embarked.notnull() == False ] )
```

Out[30]:

```
492
```

In [31]:

```
### Remove rows where there are NA values in any of the columns
```

In [32]:

```
clean_titanic_data = titanic_data.dropna()
```

In [33]:

```
len( clean_titanic_data )
```

Out[33]:

```
20
```

In [34]:

```
clean_titanic_data = titanic_data.dropna( how = "all" )
len( clean_titanic_data )
```

Out[34]:

```
1313
```

In [35]:

```
### Remove columns where all the values are NAs
```

In [36]:

```
clean_titanic_data = titanic_data.dropna( axis = 1, how = "all" )
clean_titanic_data.shape
```

Out[36]:

```
(1313, 11)
```

In [37]:

```
#titanic_data[-titanic_data.name.str.contains('Miss')]
```

Rename a column

The first column name is row.names. We can change it to rownum. As it is mostly a unique number.

In [38]:

```
titanic_data.head()
```

Out[38]:

	row.names	pclass	survived	name	age	embarked	home.dest
0	1	1st	1	Allen, Miss Elisabeth Walton	29.0000	Southampton	St Louis, MO
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chesterville, ON
2	3	1st	0	Allison, Mr Hudson Joshua Creighton	30.0000	Southampton	Montreal, PQ / Chesterville, ON
3	4	1st	0	Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)	25.0000	Southampton	Montreal, PQ / Chesterville, ON
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chesterville, ON

In [39]:

```
titanic_data.rename( columns = { 'row.names': 'rownum' },
                     inplace = True  )
```

In [40]:

```
titanic_data.head()
```

Out[40]:

	rownum	pclass	survived	name	age	embarked	home.dest	ro
0	1	1st	1	Allen, Miss Elisabeth Walton	29.0000	Southampton	St Louis, MO	B-!
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chesterville, ON	C2
2	3	1st	0	Allison, Mr Hudson Joshua Creighton	30.0000	Southampton	Montreal, PQ / Chesterville, ON	C2
3	4	1st	0	Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)	25.0000	Southampton	Montreal, PQ / Chesterville, ON	C2
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chesterville, ON	C2

Indexing and Selecting

Select first 10 rows and all the columns

In [41]:

```
first_10 = titanic_data[0:10]
```

In [42]:

```
first_10
```

Out[42]:

	rownum	pclass	survived	name	age	embarked	home.dest
0	1	1st	1	Allen, Miss Elisabeth Walton	29.0000	Southampton	St Louis, MO
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chesterville, ON
2	3	1st	0	Allison, Mr Hudson Joshua Creighton	30.0000	Southampton	Montreal, PQ / Chesterville, ON
3	4	1st	0	Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)	25.0000	Southampton	Montreal, PQ / Chesterville, ON
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chesterville, ON
5	6	1st	1	Anderson, Mr Harry	47.0000	Southampton	New York, NY
6	7	1st	1	Andrews, Miss Kornelia Theodosia	63.0000	Southampton	Hudson, NY
7	8	1st	0	Andrews, Mr Thomas, jr	39.0000	Southampton	Belfast, NI
8	9	1st	1	Appleton, Mrs Edward Dale (Charlotte Lamson)	58.0000	Southampton	Bayside, Queens, NY
9	10	1st	0	Artagaveytia, Mr Ramon	71.0000	Cherbourg	Montevideo, Uruguay

titanic_data[0:10] is same as titanic_data[:10]
Select only first 3 columns of first 10 rows

In [43]:

```
first_10_3 = titanic_data.iloc[0:10,0:3]
```

In [44]:

```
first_10_3
```

Out[44]:

	rownum	pclass	survived
0	1	1st	1
1	2	1st	0
2	3	1st	0
3	4	1st	0
4	5	1st	1
5	6	1st	1
6	7	1st	1
7	8	1st	0
8	9	1st	1
9	10	1st	0

How to access last rows

In [45]:

```
## Accesising last row
titanic_data[-1:]
```

Out[45]:

	rownum	pclass	survived	name	age	embarked	home.dest	roo
1312	1313	3rd	0	Zimmerman, Leo	NaN	NaN	NaN	Nal

In [46]:

```
last_10 = titanic_data[-10:]
```

In [47]:

```
last_10
```

Out[47]:

	rownum	pclass	survived	name	age	embarked	home.dest	roc
1303	1304	3rd	0	Yasbeck, Mr Antoni	NaN	NaN	NaN	Nal
1304	1305	3rd	1	Yasbeck, Mrs Antoni	NaN	NaN	NaN	Nal
1305	1306	3rd	0	Youssef, Mr Gerios	NaN	NaN	NaN	Nal
1306	1307	3rd	0	Zabour, Miss Hileni	NaN	NaN	NaN	Nal
1307	1308	3rd	0	Zabour, Miss Tamini	NaN	NaN	NaN	Nal
1308	1309	3rd	0	Zakarian, Mr Artun	NaN	NaN	NaN	Nal
1309	1310	3rd	0	Zakarian, Mr Maprieder	NaN	NaN	NaN	Nal
1310	1311	3rd	0	Zenn, Mr Philip	NaN	NaN	NaN	Nal
1311	1312	3rd	0	Zievens, Rene	NaN	NaN	NaN	Nal
1312	1313	3rd	0	Zimmerman, Leo	NaN	NaN	NaN	Nal

Selecting rows and columns and applying a filtering criteria

Only age, sex and pclass of passengers who have survived

In [48]:

```
titanic_data[ ( titanic_data.survived == 1 ) &
             ( titanic_data.age <= 5 ) ][['age',
                                          'sex',
                                          'pclass']][0:5]
```

Out[48]:

	age	sex	pclass
4	0.9167	male	1st
86	4.0000	male	1st
338	4.0000	female	2nd
339	1.0000	male	2nd
358	0.8333	male	2nd

Only age, sex, survived and pclass of passengers whose age are not known

In [49]:

```
titanic_data[ titanic_data.age.isnull() ][['age',
                                           'survived',
                                           'sex',
                                           'pclass']][0:5]
```

Out[49]:

	age	survived	sex	pclass
12	NaN	1	female	1st
13	NaN	1	male	1st
14	NaN	0	male	1st
29	NaN	0	male	1st
32	NaN	1	male	1st

Only age, sex, survived and pclass of passengers whose age are known

In [50]:

```
titanic_data[ -titanic_data.age.isnull() ][['age',
                                             'survived',
                                             'sex',
                                             'pclass']][0:5]
```

Out[50]:

	age	survived	sex	pclass
0	29.0000	1	female	1st
1	2.0000	0	female	1st
2	30.0000	0	male	1st
3	25.0000	0	female	1st
4	0.9167	1	male	1st

Only age, sex, survived and pclass of passengers whose age are known and have survived

In [51]:

```
titanic_data[ -titanic_data.age.isnull() &
             titanic_data.survived == 0 ][['age',
                                            'sex',
                                            'pclass']][0:5]
```

Out[51]:

	age	sex	pclass
1	2	female	1st
2	30	male	1st
3	25	female	1st
7	39	male	1st
9	71	male	1st

Removing rows with null values...

In [52]:

```
titanic_no_null = titanic_data['age'].fillna( 2 )
```

In [53]:

```
titanic_no_null[0:5]
```

Out[53]:

```
0    29.0000
1     2.0000
2    30.0000
3    25.0000
4     0.9167
Name: age, dtype: float64
```

In [54]:

```
len( titanic_no_null )
```

Out[54]:

```
1313
```

In [55]:

```
titanic_no_null = titanic_data[['age','survived',
                                'pclass','sex']].dropna()
```

Add a new column and map values of an existing column

In [56]:

```
titanic_data["gender"] = titanic_data.sex.map( lambda x:
                                              int( x == 'male') )
```

In [57]:

```
titanic_data.head()
```

Out[57]:

	rownum	pclass	survived	name	age	embarked	home.dest
0	1	1st	1	Allen, Miss Elisabeth Walton	29.0000	Southampton	St Louis, MO
1	2	1st	0	Allison, Miss Helen Loraine	2.0000	Southampton	Montreal, PQ / Chesterville, ON
2	3	1st	0	Allison, Mr Hudson Joshua Creighton	30.0000	Southampton	Montreal, PQ / Chesterville, ON
3	4	1st	0	Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)	25.0000	Southampton	Montreal, PQ / Chesterville, ON
4	5	1st	1	Allison, Master Hudson Trevor	0.9167	Southampton	Montreal, PQ / Chesterville, ON

Remove a column from dataframe

In [58]:

```
titanic_data.drop( "sex", inplace = True, axis = 1 )
```

Finding basic statistics and dawing a basic distribution plot

In [59]:

```
titanic_no_null.describe()
```

Out[59]:

	age	survived
count	633.000000	633.000000
mean	31.194181	0.443918
std	14.747525	0.497238
min	0.166700	0.000000
25%	21.000000	0.000000
50%	30.000000	0.000000
75%	41.000000	1.000000
max	71.000000	1.000000

In [60]:

```
mean_age = titanic_no_null.age.mean()
mean_age
```

Out[60]:

```
31.19418104265403
```

In [61]:

```
std_age = titanic_no_null.age.std()
std_age
```

Out[61]:

```
14.747525275652212
```

In [62]:

```
titanic_no_null[titanic_no_null.survived == 1]['age'].mean()
```

Out[62]:

```
29.873961921708187
```

In [63]:

```
titanic_no_null[titanic_no_null.survived == 1
                & titanic_no_null.sex.str.startswith( 'female' )]['age'].mean()
```

Out[63]:

```
32.43491124260355
```

In [64]:

```
titanic_no_null[titanic_no_null.survived == 1
                & titanic_no_null.sex.str.startswith( 'male' )]['age'].mean()
```

Out[64]:

```
26.201719047619044
```

In [65]:

```
titanic_no_null[titanic_no_null.survived == 0]['age'].mean()
```

Out[65]:

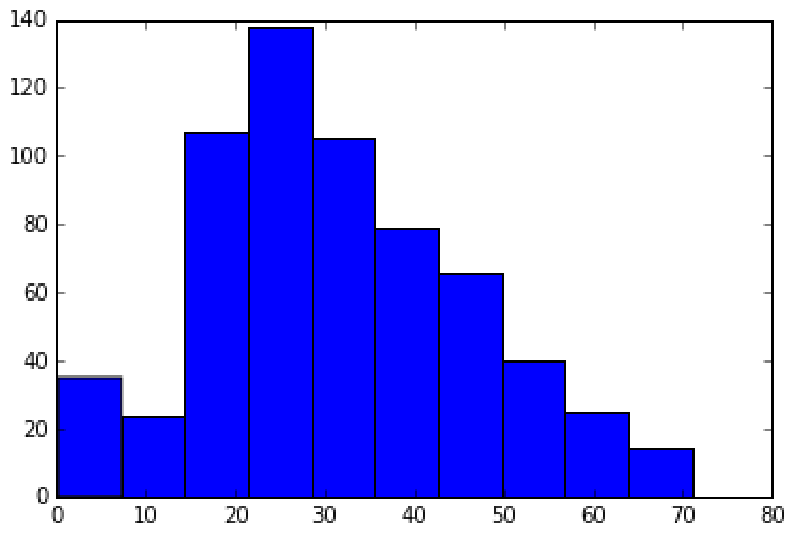
```
32.24810596590909
```

In [66]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [67]:

```
fig = plt.hist( titanic_no_null.age )
```

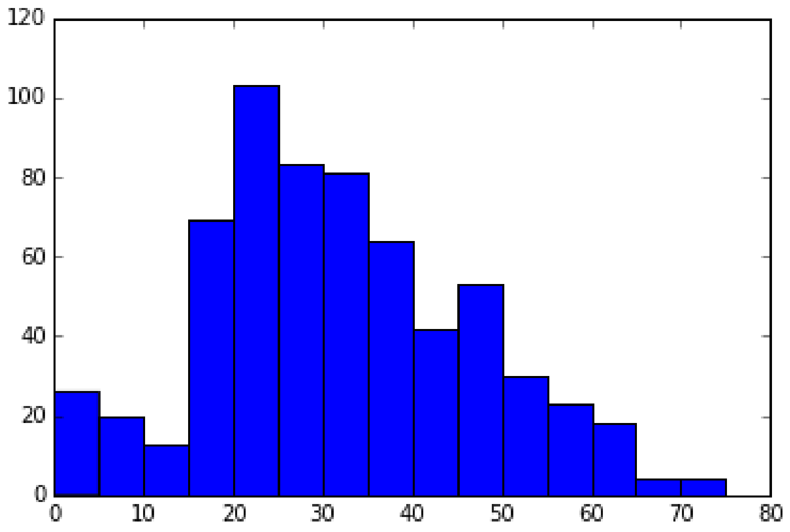


In [68]:

```
plt.hist( titanic_no_null.age, bins=16, range=(0,80))
```

Out[68]:

```
(array([  26.,   20.,   13.,   69.,  103.,   83.,   81.,   64.,   4
2.,
          53.,   30.,   23.,   18.,    4.,    4.,    0.]),
 array([  0.,   5.,  10.,  15.,  20.,  25.,  30.,  35.,  40.,  45.,
50.,
         55.,  60.,  65.,  70.,  75.,  80.]),
 <a list of 16 Patch objects>)
```

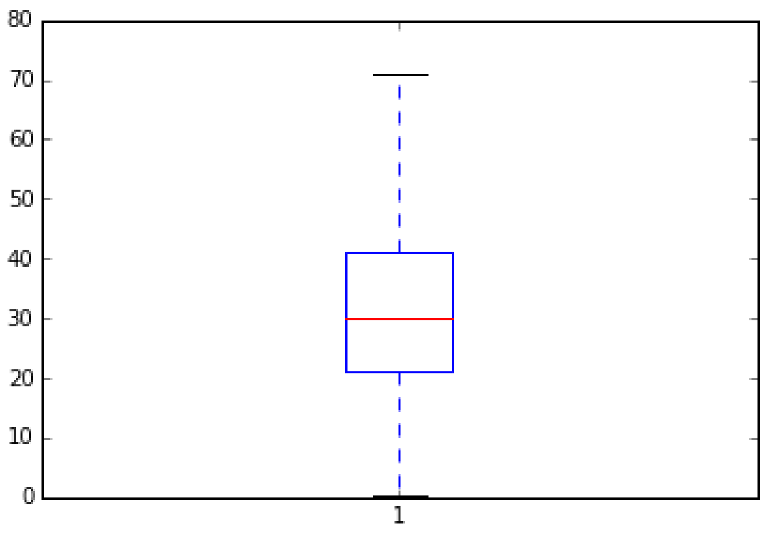


```
In [70]:
```

```
plt.boxplot( titanic_no_null.age )
```

```
Out[70]:
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x8e36438>],
 'caps': [<matplotlib.lines.Line2D at 0x8e3c6d8>,
  <matplotlib.lines.Line2D at 0x8e3ce10>],
 'fliers': [<matplotlib.lines.Line2D at 0x8e42da0>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x8e425f8>],
 'whiskers': [<matplotlib.lines.Line2D at 0x8e366d8>,
  <matplotlib.lines.Line2D at 0x8e36ef0>]}
```



Make note of lessons learnt in this exercise