Adding attributes

```
In [3]: # importing pandas as pd
         import pandas as pd
         # Creating the DataFrame
         df = pd.DataFrame({'Date':['10/2/2011', '11/2/2011', '12/2/2011', '13/2/2011'],
                           'Event':['Music', 'Poetry', 'Theatre', 'Comedy'],
                           'Cost':[10000, 5000, 15000, 2000]})
         # Print the dataframe
         print(df)
                Date Event Cost
         0 10/2/2011 Music 10000
         1 11/2/2011 Poetry 5000
         2 12/2/2011 Theatre 15000
         3 13/2/2011 Comedy 2000
In [4]: #using apply function to create a new column
         df['Discounted_Price'] = df.apply(lambda row: row.Cost * 0.9, axis = 1)
         # Print the DataFrame after addition of new column
         print(df)
                Date Event Cost Discounted_Price
         0 10/2/2011 Music 10000
                                              9000.0
         1 11/2/2011 Poetry
                              5000
                                              4500.0
         2 12/2/2011 Theatre 15000
                                              13500.0
         3 13/2/2011
                      Comedy
                               2000
                                               1800.0
In [28]: df = pd.DataFrame({'Name':['John','Ted','Dove','Brad','Rex'],
                           'Salary':[44000, 35000, 75000, 20000,6000]})
         # Print the dataframe
         print(df)
           Name Salary
         0 John
                  44000
         1
           Ted
                35000
         2 Dove 75000
         3 Brad 20000
           Rex 6000
In [29]: def salary_stats(value):
             if value < 10000:
                return "very low"
             elif 10000 <= value < 25000:
                return "low"
             elif 25000 <= value < 40000:
                return "average"
             elif 40000 <= value < 50000:
                return "better"
             elif value >= 50000:
                return "very good"
         df['salary stats'] = df['Salary'].map(salary stats)
         df
```

```
Out[29]:
              Name Salary salary_stats
                John
                      44000
                                    better
                      35000
                 Ted
                                  average
               Dove
                      75000
                                very good
           3
                      20000
                Brad
                                      low
                 Rex
                        6000
                                 very low
```

```
Education
Out[85]:
              Name
           0
                      High School
                   Α
                          Masters
           2
                   C
                        Doctorate
           3
                        Bachelors
           4
                   Ε
                          Masters
                   F High School
```

Binary encoding

```
education_data = pd.get_dummies(data.Education)
In [86]:
         print(education_data)
            Bachelors Doctorate High School Masters
         0
                                0
         1
                    0
                                0
                                             0
                                                       1
         2
                     0
                                1
                                             0
                                                       0
         3
                     1
                                0
                                             0
                                                       0
         4
                                             0
                                                       1
         5
                                0
```

Ranking Transformation

```
In [80]:
    education_map = {
        'High School' : 1,
        'Bachelors' : 2,
        'Masters': 3,
        'Doctorate': 4
    }
    education_data = data['Education'].map(education_map)
    data['Education'] = education_data
    data
```

ut[80]:		Name	Education
	0	А	1
	1	В	3
	2	С	4
	3	D	2
	4	Е	3
	5	F	1

```
In [84]: education_map = {
    'High School' : 12,
    'Bachelors' : 16,
    'Masters': 18,
    'Doctorate': 21
}
education_data = data['Education'].map(education_map)
data['Education'] = education_data
data
```

Out[84]:		Name	Education
	0	А	12
	1	В	18
	2	С	21
	3	D	16
	4	Е	18
	5	F	12

Adding data objects- rows

```
In [71]: df.loc[len(df.index)]=['Hruthvik', 15000, 'low']
    df
```

Out[71]:		Name	Salary	salary_stats
	0	John	44000	better
	1	Ted	35000	average
	2	Dove	75000	very good
	3	Brad	20000	low
	4	Rex	6000	very low
	5	Hruthvik	15000	low
	6	Hruthvik	15000	low
	7	Hruthvik	15000	low

Combining two dataframes

```
In [48]:
         import pandas as pd
         d1 = {'Name': ['Pankaj', 'Meghna', 'Lisa'], 'Country': ['India', 'India', 'USA'],
         df1 = pd.DataFrame(d1)
         print('DataFrame 1:\n', df1,'\n')
         df2 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']})
         print('DataFrame 2:\n', df2,'\n')
         df3 = pd.DataFrame({'Name': ['Priya'], 'Country': ['India'], 'Role': ['C00']})
         print('DataFrame 3:\n', df3,'\n')
         DataFrame 1:
               Name Country Role
         0 Pankaj
                     India CEO
         1 Meghna
                     India CTO
         2
                       USA CTO
              Lisa
         DataFrame 2:
             TD
                   Name
             1 Pankaj
         1
             2 Anupam
         2
             3
                  Amit
         DataFrame 3:
              Name Country Role
         0 Priya
                    India COO
         same cols df = pd.concat([df1,df3],ignore index=True)
In [49]:
         same_cols_df
Out[49]:
              Name Country
                            Role
                             CEO
             Pankaj
                       India
         1 Meghna
                       India
                             CTO
         2
                       USA
                            CTO
               Lisa
               Priya
                       India COO
         a_df=df1.append(df2, ignore_index=True)
In [50]:
         C:\Users\YASH\AppData\Local\Temp\ipykernel 13020\2048347772.py:1: FutureWarning: T
         he frame.append method is deprecated and will be removed from pandas in a future v
         ersion. Use pandas.concat instead.
           a_df=df1.append(df2, ignore_index=True)
Out[50]:
              Name Country Role
                                   ID
              Pankaj
                       India
                             CEO NaN
         1 Meghna
                       India
                            CTO
                                 NaN
         2
                Lisa
                        USA
                            CTO
                                  NaN
                       NaN
         3
              Pankaj
                             NaN
                                   1.0
         4 Anupam
                       NaN
                             NaN
                                   2.0
```

NaN

NaN

5

Amit

3.0

```
c_df = pd.concat([df1,df2],ignore_index=True)
In [51]:
              Name Country Role
Out[51]:
              Pankaj
                       India
                             CEO NaN
         1 Meghna
                       India
                            CTO NaN
         2
                Lisa
                        USA
                            CTO NaN
         3
              Pankaj
                        NaN
                            NaN
                                   1.0
         4 Anupam
                        NaN
                            NaN
                                   2.0
               Amit
                        NaN NaN
                                   3.0
```

Default Merging - inner join

```
In [52]: df_merged = df1.merge(df2)
    print('Result:\n', df_merged)

Result:
        Name Country Role ID
        0 Pankaj India CEO 1
```

Merging DataFrames with Left, Right, and Outer Join

```
print('Result Left Join:\n', df1.merge(df2, how='left'))
print('Result Right Join:\n', df1.merge(df2, how='right'))
print('Result Outer Join:\n', df1.merge(df2, how='outer'))
Result Left Join:
     Name Country Role
0 Pankaj India CEO 1.0
           India CTO NaN
1 Meghna
            USA CTO NaN
    Lisa
Result Right Join:
     Name Country Role ID
0 Pankaj
         India CEO
1 Anupam
            NaN NaN
                       2
    Amit
             NaN NaN
Result Outer Join:
     Name Country Role
0 Pankaj India CEO 1.0
1 Meghna
         India CTO NaN
            USA CTO NaN
2
   Lisa
3 Anupam
            NaN NaN
                      2.0
    Amit
            NaN NaN 3.0
```

Merging DataFrame on Specific Columns

```
df2 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']})
print(df1.merge(df2, on='ID'))
print('\n',df1.merge(df2, on='Name'))
```

```
ID Name_x Country Role Name_y
0  1 Pankaj India CEO Pankaj
1  2 Meghna India CTO Anupam
2  3 Lisa USA CTO Amit

ID_x Name Country Role ID_y
0  1 Pankaj India CEO 1
```

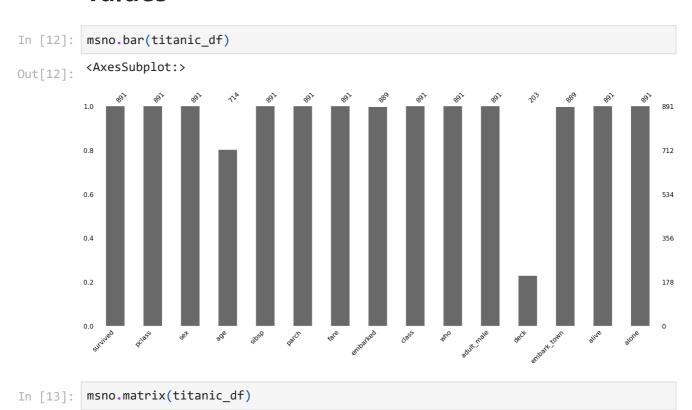
```
In [7]:
          # Package imports
          import pandas as pd
          import missingno as msno
          %matplotlib inline
 In [8]:
          #Importing the required dataset
          titanic_df = pd.read_csv("titanic.csv")
          titanic_df
 Out[8]:
               survived
                        pclass
                                            sibsp parch
                                                                  embarked
                                                                               class
                                                                                       who
                                                                                            adult male
                                  sex
                                        age
            0
                                                           7.2500
                                                                          S
                      0
                             3
                                       22.0
                                                       0
                                                                               Third
                                                                                                   Tru€
                                 male
                                                                                       man
                                       38.0
                                                         71.2833
                                                                          C
            1
                               female
                                                                               First woman
                                                                                                  False
                                                           7.9250
                                                                          S
            2
                      1
                                       26.0
                                                0
                             3
                               female
                                                       0
                                                                               Third
                                                                                                  Fals€
                                                                                    woman
                               female
                                       35.0
                                                          53.1000
                                                                               First
                                                                                    woman
                                                                                                  False
                      0
                                                                          S
            4
                             3
                                       35.0
                                                0
                                                       0
                                                           8.0500
                                                                               Third
                                 male
                                                                                                   Tru€
                                                                                       man
          886
                      0
                             2
                                       27.0
                                                0
                                                       0 13.0000
                                                                          S Second
                                 male
                                                                                       man
                                                                                                   Tru€
          887
                               female
                                       19.0
                                                          30.0000
                                                                          S
                                                                               First
                                                                                    woman
                                                                                                  Fals€
          888
                      0
                                                       2 23.4500
                                                                          S
                                                                                                  Fals€
                             3
                               female
                                      NaN
                                                1
                                                                               Third woman
          889
                                       26.0
                                                         30.0000
                                                                          C
                                                                               First
                                                                                                   Tru€
                                 male
                                                                                       man
          890
                      0
                             3
                                                0
                                                                         Q
                                       32.0
                                                       0
                                                          7.7500
                                                                               Third
                                 male
                                                                                                   Tru€
                                                                                       man
         891 rows × 15 columns
          titanic_df.info()
In [10]:
          <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
                              714 non-null
                                                float64
           3
               age
                              891 non-null
                                                int64
           4
               sibsp
           5
               parch
                              891 non-null
                                                int64
                                                float64
           6
               fare
                              891 non-null
           7
               embarked
                              889 non-null
                                                object
           8
               class
                              891 non-null
                                                object
           9
               who
                              891 non-null
                                                object
                              891 non-null
                                                bool
           10
               adult_male
                                                object
           11
               deck
                              203 non-null
                                                object
           12
               embark_town
                              889 non-null
           13
               alive
                              891 non-null
                                                object
           14
               alone
                              891 non-null
                                                bool
          dtypes: bool(2), float64(2), int64(4), object(7)
          memory usage: 92.4+ KB
```

titanic_df.isnull()

In [11]:

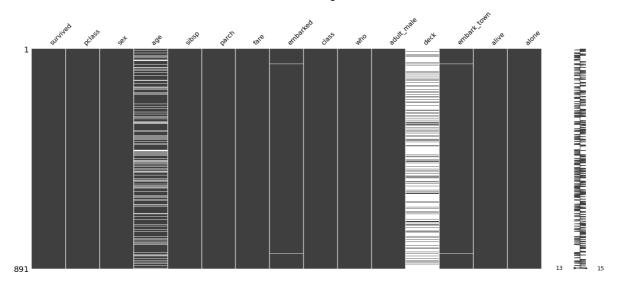
Out[11]:	s	urvived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
	0	False	False	False	False	False	False	False	False	False	False	False	True
	1	False	False	False	False	False	False	False	False	False	False	False	False
	2	False	False	False	False	False	False	False	False	False	False	False	True
	3	False	False	False	False	False	False	False	False	False	False	False	False
	4	False	False	False	False	False	False	False	False	False	False	False	True
	•••												
	886	False	False	False	False	False	False	False	False	False	False	False	True
	887	False	False	False	False	False	False	False	False	False	False	False	False
	888	False	False	False	True	False	False	False	False	False	False	False	True
	889	False	False	False	False	False	False	False	False	False	False	False	False
	890	False	False	False	False	False	False	False	False	False	False	False	True
	891 rov	ws × 15	column	S									

Using Missingno to visualize missing values



<AxesSubplot:>

Out[13]:



Deleting the entire row

```
titanic_df.isnull().sum()
In [14]:
          survived
Out[14]:
                            0
          pclass
                            0
          sex
          age
                          177
          sibsp
                            0
          parch
                            0
          fare
                            0
          embarked
                            2
          class
                            0
          who
                            0
                            0
          adult_male
          deck
                          688
          embark_town
                            2
          alive
                            0
          alone
                            0
          dtype: int64
In [15]: df = titanic_df.dropna(axis=0)
          df.isnull().sum()
          survived
Out[15]:
                          0
          pclass
                          0
          sex
                          0
          age
          sibsp
                          0
          parch
          fare
                          0
          embarked
                          0
          class
                          0
          who
                          0
                          0
          adult_male
                          0
          deck
                          0
          embark_town
                          0
          alive
          alone
          dtype: int64
In [16]:
          df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 182 entries, 1 to 889
Data columns (total 15 columns):
# Column Non-Null Count Dtype
                    -----
--- -----
0
    survived 182 non-null
                                       int64
1 pclass 182 non-null int64
                  182 non-null object
 2 sex
   age
                  182 non-null float64
 3
4 sibsp 182 non-null int64
5 parch 182 non-null int64
6 fare 182 non-null float64
7 embarked 182 non-null object
8 class 182 non-null object
9 who 182 non-null object
10 adult_male 182 non-null bool
                    182 non-null object
 11 deck
 12 embark_town 182 non-null object
13 alive 182 non-null object
14 alone 182 non-null bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 20.3+ KB
```

Deleting the entire column

```
In [17]: titanic_df.columns
       Out[17]:
             'alive', 'alone'],
            dtype='object')
       df = titanic_df.drop(['deck'],axis=1)
       df.isnull().sum()
       survived
Out[18]:
       pclass
       sex
                   177
       age
       sibsp
       parch
       fare
       embarked
       class
       who
       adult male
       embark_town
                    2
       alive
                    0
       alone
       dtype: int64
```

Imputing the Missing Value

There are different ways of replacing the missing values.

Replacing With Arbitrary Value

If you can make an educated guess about the missing value then you can replace it with some arbitrary value using the following code.

```
titanic_df['deck'].unique()
In [20]:
           array([nan, 'C', 'E', 'G', 'D', 'A', 'B', 'F'], dtype=object)
Out[20]:
           titanic_df['deck'] = titanic_df['deck'].fillna('C')
In [21]:
           titanic_df['deck'].isnull().sum() #missing values replaced
Out[23]:
In [24]:
           titanic_df
Out[24]:
                                                                     embarked
                survived pclass
                                               sibsp
                                                     parch
                                                                fare
                                                                                   class
                                                                                            who
                                                                                                adult_male
                                    sex
                                          age
             0
                                                              7.2500
                                                                              S
                       0
                              3
                                   male
                                          22.0
                                                   1
                                                          0
                                                                                   Third
                                                                                            man
                                                                                                        True
             1
                       1
                                         38.0
                                                          0 71.2833
                                                                              C
                                 female
                                                                                    First
                                                                                         woman
                                                                                                       False
             2
                       1
                              3
                                 female
                                          26.0
                                                   0
                                                          0
                                                              7.9250
                                                                              S
                                                                                   Third
                                                                                         woman
                                                                                                       Fals€
                                                                              S
             3
                       1
                                         35.0
                                                          0
                                                             53.1000
                                 female
                                                                                    First
                                                                                         woman
                                                                                                       False
             4
                       0
                              3
                                          35.0
                                                   0
                                                          0
                                                              8.0500
                                                                              S
                                                                                   Third
                                   male
                                                                                            man
                                                                                                        Tru€
           886
                       0
                              2
                                         27.0
                                                             13.0000
                                                                              S Second
                                   male
                                                   0
                                                                                                        Tru€
                                                                                            man
           887
                                         19.0
                                                          0 30.0000
                                                                              S
                                 female
                                                                                    First
                                                                                         woman
                                                                                                       Fals€
           888
                                                          2 23.4500
                                                                              S
                                                                                                       Fals€
                              3
                                 female
                                         NaN
                                                   1
                                                                                   Third
                                                                                         woman
           889
                                         26.0
                                                          0 30.0000
                                                                              C
                                                   0
                                                                                    First
                                                                                                        True
                                   male
                                                                                            man
           890
                       0
                              3
                                   male
                                          32.0
                                                   0
                                                              7.7500
                                                                             Q
                                                                                   Third
                                                                                            man
                                                                                                        Tru€
          891 rows × 15 columns
```

Replacing With Mean

This is the most common method of imputing missing values of numeric columns.

```
In [25]: mean = titanic_df['age'].mean()
    print(mean)
    #Replace the missing values for numerical columns with mean
    titanic_df['age'] = titanic_df['age'].fillna(mean)
    titanic_df['age']
```

29.69911764705882

```
22.000000
Out[25]:
          1
                 38.000000
          2
                 26.000000
                 35.000000
                 35.000000
                   . . .
          886
                 27.000000
          887
                 19.000000
          888
                 29.699118
          889
                 26.000000
          890
                 32.000000
          Name: age, Length: 891, dtype: float64
```

Replacing With Mode

Mode is the most frequently occurring value. It is used in the case of categorical features.

```
titanic_df = pd.read_csv("titanic.csv")
In [26]:
         #Replace the missing values for categorical columns with mode
         mode = titanic_df['deck'].mode()[0]
         print(mode)
         titanic_df['deck'] = titanic_df['deck'].fillna(mode)
         C
         titanic_df['deck']
In [27]:
                C
Out[27]:
                C
         2
                C
         3
                C
                C
         886
                C
         887
                В
         888
                C
         889
                C
         Name: deck, Length: 891, dtype: object
```

Replacing With Median

Median is the middlemost value. It's better to use the median value for imputation in the case of outliers.

```
In [28]: titanic_df['age']= titanic_df['age'].fillna(titanic_df['age'].median())
    titanic_df['age']
```

```
22.0
Out[28]:
          1
                  38.0
          2
                  26.0
          3
                  35.0
          4
                  35.0
                  . . .
          886
                  27.0
          887
                  19.0
          888
                  28.0
          889
                  26.0
          890
                  32.0
          Name: age, Length: 891, dtype: float64
```

Forward and backward filling of missing values

```
titanic_df = pd.read_csv("titanic2.csv")
In [29]:
           titanic_df
Out[29]:
                                                                     embarked
                                                                                   class
                survived
                          pclass
                                               sibsp
                                                                fare
                                                                                           who adult male
                                          age
                                                     parch
                                    sex
             0
                     0.0
                              3
                                   male
                                         22.0
                                                          0
                                                              7.2500
                                                                             S
                                                                                   Third
                                                                                           man
                                                                                                        True
             1
                                                                             C
                     1.0
                                         38.0
                                                          0
                                                            71.2833
                                 female
                                                                                   First
                                                                                         woman
```

Fals€ 2 NaN female NaN 7.9250 S Third Fals€ woman S 3 NaN NaN 53.1000 female First woman Fals€ 4 0.0 3 male 35.0 0 8.0500 Third True man 886 0.0 2 male 27.0 13.0000 S Second man Tru€ 887 30.0000 1.0 female 19.0 First woman False 888 S 0.0 female NaN 23.4500 Third woman Fals€ 889 1.0 26.0 30.0000 First True male man

7.7500

Q

Third

man

891 rows × 15 columns

0.0

3

male

32.0

890

```
In [30]: new_df = titanic_df.fillna(method="ffill")
    new_df
```

Tru€

12/15/22, 4:06 PM

MissingValues Out[30]: survived pclass age sibsp parch fare embarked class who adult_male sex 0.0 0 3 male 22.0 1 0 7.2500 S Third man True 1 1.0 38.0 1 0 71.2833 C False 1 female First woman S 2 1.0 3 female 38.0 0 0 7.9250 Third woman False 3 1.0 38.0 1 0 53.1000 S First woman False female 4 0.0 3 male 35.0 0 0 8.0500 S Third man True 0.0 2 0 S Second 886 male 27.0 0 13.0000 man True S 887 1.0 19.0 0 0 30.0000 1 female First woman False 888 0.0 S female 19.0 1 2 23.4500 Third woman **False** C 889 1.0 male 26.0 0 0 30.0000 First True man 890 0.0 3 male 32.0 0 Q Third 7.7500 man True 891 rows × 15 columns 4 • In [31]: new_df = titanic_df.fillna(method="ffill",limit=1) new_df Out

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0.0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1.0	1	female	38.0	1	0	71.2833	С	First	woman	False
2	1.0	3	female	38.0	0	0	7.9250	S	Third	woman	False
3	NaN	1	female	NaN	1	0	53.1000	S	First	woman	False
4	0.0	3	male	35.0	0	0	8.0500	S	Third	man	Tru€
•••		•••									
886	0.0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1.0	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0.0	3	female	19.0	1	2	23.4500	S	Third	woman	False
889	1.0	1	male	26.0	0	0	30.0000	С	First	man	True
890	0.0	3	male	32.0	0	0	7.7500	Q	Third	man	Tru€
	1 2 3 4 886 887 888 889	0 0.0 1 1.0 2 1.0 3 NaN 4 0.0 886 0.0 887 1.0 888 0.0 889 1.0	0 0.0 3 1 1.0 1 2 1.0 3 3 NaN 1 4 0.0 3 886 0.0 2 887 1.0 1 888 0.0 3 889 1.0 1	0 0.0 3 male 1 1.0 1 female 2 1.0 3 female 3 NaN 1 female 4 0.0 3 male 886 0.0 2 male 887 1.0 1 female 888 0.0 3 female 889 1.0 1 male	0 0.0 3 male 22.0 1 1.0 1 female 38.0 2 1.0 3 female 38.0 3 NaN 1 female NaN 4 0.0 3 male 35.0 886 0.0 2 male 27.0 887 1.0 1 female 19.0 888 0.0 3 female 19.0 889 1.0 1 male 26.0	0 0.0 3 male 22.0 1 1 1.0 1 female 38.0 1 2 1.0 3 female 38.0 0 3 NaN 1 female NaN 1 4 0.0 3 male 35.0 0 886 0.0 2 male 27.0 0 887 1.0 1 female 19.0 0 888 0.0 3 female 19.0 1 889 1.0 1 male 26.0 0	0 0.0 3 male 22.0 1 0 1 1.0 1 female 38.0 1 0 2 1.0 3 female 38.0 0 0 3 NaN 1 female NaN 1 0 4 0.0 3 male 35.0 0 0 886 0.0 2 male 27.0 0 0 887 1.0 1 female 19.0 0 0 888 0.0 3 female 19.0 1 2 889 1.0 1 male 26.0 0 0	0 0.0 3 male 22.0 1 0 7.2500 1 1.0 1 female 38.0 1 0 71.2833 2 1.0 3 female 38.0 0 0 7.9250 3 NaN 1 female NaN 1 0 53.1000 4 0.0 3 male 35.0 0 0 8.0500 886 0.0 2 male 27.0 0 0 13.0000 887 1.0 1 female 19.0 0 0 30.0000 888 0.0 3 female 19.0 1 2 23.4500 889 1.0 1 male 26.0 0 0 30.0000	0 0.0 3 male 22.0 1 0 7.2500 S 1 1.0 1 female 38.0 1 0 71.2833 C 2 1.0 3 female 38.0 0 0 7.9250 S 3 NaN 1 female NaN 1 0 53.1000 S 4 0.0 3 male 35.0 0 0 8.0500 S 886 0.0 2 male 27.0 0 0 13.0000 S 887 1.0 1 female 19.0 0 30.0000 S 888 0.0 3 female 19.0 1 2 23.4500 S 889 1.0 1 male 26.0 0 0 30.0000 C	0 0.0 3 male 22.0 1 0 7.2500 S Third 1 1.0 1 female 38.0 1 0 71.2833 C First 2 1.0 3 female 38.0 0 0 7.9250 S Third 3 NaN 1 female NaN 1 0 53.1000 S First 4 0.0 3 male 35.0 0 0 8.0500 S Third 886 0.0 2 male 27.0 0 0 13.0000 S Second 887 1.0 1 female 19.0 1 2 23.4500 S Third 888 0.0 3 female 19.0 0 30.0000 C First 889 1.0 1 male 26.0 0 30.0000 C Fi	0 0.0 3 male 22.0 1 0 7.2500 S Third man 1 1.0 1 female 38.0 1 0 71.2833 C First woman 2 1.0 3 female 38.0 0 0 7.9250 S Third woman 3 NaN 1 female NaN 1 0 53.1000 S First woman 4 0.0 3 male 35.0 0 0 8.0500 S Third man 886 0.0 2 male 27.0 0 0 13.0000 S Second man 887 1.0 1 female 19.0 0 30.0000 S Third woman 888 0.0 3 female 19.0 1 2 23.4500 S Third woman 889 1.0

891 rows × 15 columns

```
new df = titanic df.fillna(method="bfill")
In [33]:
         new_df
```

\cap	14-	ГЭ	2	٦.
U	ЛL	L⊃	2	Ŀ

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0.0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1.0	1	female	38.0	1	0	71.2833	С	First	woman	False
2	0.0	3	female	35.0	0	0	7.9250	S	Third	woman	False
3	0.0	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0.0	3	male	35.0	0	0	8.0500	S	Third	man	True
•••											
886	0.0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1.0	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0.0	3	female	26.0	1	2	23.4500	S	Third	woman	False
889	1.0	1	male	26.0	0	0	30.0000	С	First	man	True
890	0.0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns

Numerosity Data Reduction

Random sampling

Example - Random sampling to speed up tuning

```
import numpy as np
In [1]:
         import matplotlib.pyplot as plt
         import pandas as pd
         import seaborn as sns
In [4]:
         customer_df = pd.read_csv('Customer Churn.csv')
         print(customer_df.shape)
         print(customer_df.Churn.value_counts())
         (3150, 9)
         0
               2655
                495
         Name: Churn, dtype: int64
         customer_df_rs = customer_df.sample(1000, random_state=1)
In [5]:
         y=customer df rs['Churn']
         Xs = customer_df_rs.drop(columns=['Churn'])
         print(customer_df_rs.shape)
         (1000, 9)
In [6]:
         customer_df_rs
Out[6]:
                                                                                Distinct
                  Call
                                   Subscription Seconds Frequency
                                                                   Frequency
                       Complains
                                                                                 Called
                                                                                        Status Churn
               Failure
                                       Length
                                                 of Use
                                                            of use
                                                                       of SMS
                                                                               Numbers
         2001
                    0
                                0
                                            37
                                                      0
                                                                 0
                                                                            0
                                                                                      0
                                                                                             0
                                                                                                    0
          943
                    0
                                0
                                            24
                                                   2515
                                                                50
                                                                                     23
                                                                                                    0
                                                                           33
                                                                                     19
                                                                                             0
         1611
                    4
                                0
                                            37
                                                   2048
                                                                54
                                                                                                    0
          403
                                                   4018
                                                                                     30
                    8
                                0
                                            35
                                                                58
                                                                            0
                                                                                                    0
         1301
                    0
                                0
                                                                 9
                                                                           15
                                                                                      8
                                                                                             0
                                                                                                    0
                                           43
                                                    273
          446
                                                                                     35
                   10
                                0
                                            12
                                                   8753
                                                               163
                                                                           62
                                                                                             1
                                                                                                    0
         2182
                    8
                                0
                                            40
                                                                                      6
                                                                                                    0
                                                    703
                                                                13
                                                                           16
          709
                    5
                                0
                                            38
                                                   4325
                                                                82
                                                                            0
                                                                                     22
                                                                                             1
                                                                                                    0
         1721
                    5
                                0
                                            35
                                                   6603
                                                                70
                                                                          110
                                                                                     38
                                                                                             1
                                                                                                    0
                    5
                                            38
                                                   2043
                                                                40
                                                                           37
                                                                                     33
                                                                                             0
         1227
                                                                                                    1
        1000 rows × 9 columns
         print(customer_df_rs.Churn.value_counts())
```

856144

Name: Churn, dtype: int64

Stratified sampling

Example – Stratified sampling for imbalanced dataset

```
n,s=len(customer_df),1000
In [59]:
          print(n,s)
          r = s/n
          print('Ratio of each Churn class in sample:',r)
          sample_df = customer_df.groupby('Churn').apply(lambda sdf: sdf.sample(round(len(sd-
          print(sample_df.Churn.value_counts())
          3150 1000
          Ratio of each Churn class in sample: 0.31746031746031744
               843
          1
               157
          Name: Churn, dtype: int64
          customer_df.Churn.value_counts().plot.bar()
In [60]:
          <AxesSubplot:>
Out[60]:
          2500
          2000
          1500
          1000
           500
             0
In [61]:
          sample_df.Churn.value_counts().plot.bar()
          <AxesSubplot:>
Out[61]:
          800
          700
          600
          500
```

Random Over/Under-sampling

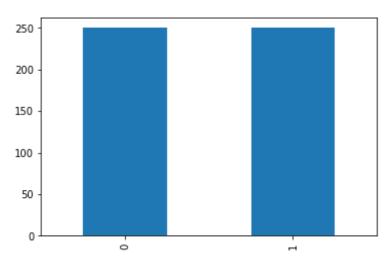
```
In [65]: n,s=len(customer_df),500
sample_df = customer_df.groupby('Churn').apply(lambda sdf: sdf.sample(250))
print(sample_df.Churn.value_counts())
```

0 2501 250

Name: Churn, dtype: int64

In [66]: sample_df.Churn.value_counts().plot.bar()

Out[66]: <AxesSubplot:>



In [67]: sample_df

_			-
() i	11	167	
- 01	<i>1</i> L	10/	

0			Call Failure	Complains	Subscription Length	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Status
	Churn									
	0	1815	0	0	30	6195	74	171	23	1
		410	4	0	35	5738	87	0	7	1
		1455	4	0	33	3290	68	14	21	1
		2086	0	0	33	1360	38	31	18	0
		506	0	0	38	1760	29	264	3	1
	•••	•••								•••
	1	2178	8	0	40	498	11	12	6	1
		1599	0	0	7	0	0	0	0	1
		1476	2	1	30	2505	27	0	9	0
		1382	0	1	28	0	0	0	0	1
		368	2	0	40	180	8	11	5	0

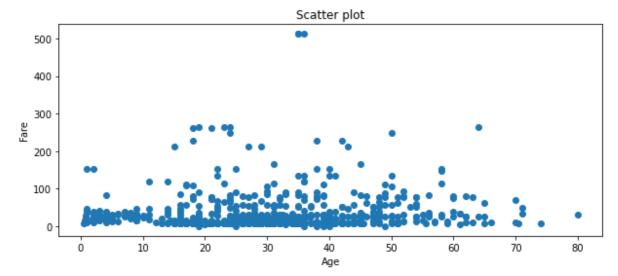
500 rows × 9 columns

Outliers Detection and handling

```
#Importing the necessary Libraries
In [2]:
          import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import matplotlib.cm as cm
          titanic_df = pd.read_csv("titanic.csv")
In [3]:
          titanic_df
Out[3]:
               survived
                         pclass
                                         age
                                             sibsp
                                                     parch
                                                               fare
                                                                     embarked
                                                                                  class
                                                                                           who
                                                                                                 adult_male
            0
                                                             7.2500
                      0
                             3
                                         22.0
                                                                             S
                                                                                  Third
                                  male
                                                         0
                                                                                                       Tru€
                                                                                           man
            1
                                female
                                         38.0
                                                            71.2833
                                                                                   First
                                                                                        woman
                                                                                                       Fals€
            2
                      1
                             3
                                female
                                        26.0
                                                  0
                                                         0
                                                             7.9250
                                                                             S
                                                                                  Third
                                                                                         woman
                                                                                                       Fals€
                                female
                                         35.0
                                                            53.1000
                                                                             S
                                                                                   First
                                                                                                       Fals€
                                                                                         woman
            4
                      0
                             3
                                        35.0
                                                  0
                                                         0
                                                             8.0500
                                                                             S
                                                                                  Third
                                  male
                                                                                                       Tru€
                                                                                           man
          886
                      0
                             2
                                         27.0
                                                            13.0000
                                                                               Second
                                  male
                                                  0
                                                                             S
                                                                                                       Tru€
                                                                                           man
          887
                                female
                                        19.0
                                                            30.0000
                                                                                   First
                                                                                        woman
                                                                                                       Fals€
          888
                      0
                             3
                                female NaN
                                                         2 23.4500
                                                                             S
                                                                                  Third
                                                                                                       Fals€
                                                                                        woman
          889
                                  male
                                        26.0
                                                         0 30.0000
                                                                                   First
                                                                                           man
                                                                                                       Tru€
          890
                      0
                             3
                                        32.0
                                                  0
                                                             7.7500
                                                                             0
                                                                                  Third
                                  male
                                                         0
                                                                                                       True
                                                                                           man
         891 rows × 15 columns
```

Scatter plot to detect outliers

```
fig,ax = plt.subplots(figsize=(10,4))
ax.scatter(titanic_df['age'],titanic_df['fare'])
ax.set_xlabel('Age')
ax.set_ylabel('Fare')
plt.title("Scatter plot")
plt.show()
```



Box plot to detect outliers

```
In [6]:
       titanic_df['age'].plot(kind='box')
        <AxesSubplot:>
Out[6]:
         80
         70
         60
         50
         40
         30
         20
         10
         0
                                  age
In [7]:
        # finding the 1st quartile
        q1 = titanic_df["age"].quantile(0.25)
        # finding the 3rd quartile
        q3 = titanic df['age'].quantile(0.75)
        # finding the iqr region
        iqr = q3-q1
        # finding upper and lower whiskers
        upper_bound = q3+(1.5*iqr)
        lower_bound = q1-(1.5*iqr)
In [8]:
        age_arr = titanic_df["age"]
        outliers = age_arr[(age_arr <= lower_bound) | (age_arr >= upper_bound)]
        print('The following are the outliers in the boxplot of age:\n',outliers)
```

```
The following are the outliers in the boxplot of age:
33
        66.0
54
       65.0
96
       71.0
116
       70.5
280
       65.0
456
       65.0
493
       71.0
       80.0
630
       70.0
672
745
       70.0
851
       74.0
Name: age, dtype: float64
```

Histogram plot to detect outliers

```
In [9]:
         titanic_df['fare'].plot(kind='hist')
         <AxesSubplot:ylabel='Frequency'>
Out[9]:
            700
            600
            500
            400
            300
            200
            100
              0
                          100
                                    200
                                             300
                                                                500
                                                       400
```

Remove data objects with outliers

```
In [10]: upperIndex = titanic_df[titanic_df['age']>upper_bound].index
    titanic_df.drop(upperIndex,inplace=True)
    lowerIndex = titanic_df[titanic_df['age']<lower_bound].index
    titanic_df.drop(lowerIndex,inplace=True)
    titanic_df.info()</pre>
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 880 entries, 0 to 890
Data columns (total 15 columns):
                 Non-Null Count Dtype
    Column
    -----
                 -----
---
                                ----
0
    survived
                 880 non-null
                                int64
               880 non-null int64
1
   pclass
               880 non-null object
    sex
3
                703 non-null
                               float64
    age
   sibsp
               880 non-null
                               int64
   parch 880 non-null int64
fare 880 non-null float64
embarked 878 non-null object
class 880 non-null object
    parch
6
7
8 class
              880 non-null object
   who
9
10 adult_male 880 non-null
                               bool
11 deck
                198 non-null object
12 embark_town 878 non-null object
13 alive 880 non-null
                                object
14 alone
                880 non-null
                                bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 98.0+ KB
```

Replacing outliers with upper and lower cap:

Upper cap is 90% Lower cap is 1%

```
In [11]: titanic_df = pd.read_csv("titanic.csv")
In [12]:
         #upper and Lower cap
         # Winzorization method
         fare_arr = titanic_df["fare"]
         upper_cap = np.percentile(fare_arr,1)
         lower_cap = np.percentile(fare_arr,99)
         outliers = fare_arr[(fare_arr < upper_cap) | (fare_arr > lower_cap)]
         print('The following are the outliers in the boxplot of fare:\n',outliers)
         The following are the outliers in the boxplot of fare:
          27
                 263.0000
         88
                263.0000
         258
                512.3292
         311
                262.3750
         341
                263.0000
         438
                263.0000
         679
                512.3292
         737
                512.3292
         742
                262.3750
         Name: fare, dtype: float64
In [13]:
         for i in titanic_df['fare']:
              if i<lower_bound :</pre>
                  titanic_df['fare'] = titanic_df['fare'].replace(i,lower_cap)
              elif i>upper_bound :
                  titanic_df['fare'] = titanic_df['fare'].replace(i,upper_cap)
         titanic df.info()
In [14]:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
    Column
                Non-Null Count Dtype
--- -----
                -----
                               ----
   survived
0
                891 non-null
                                int64
               891 non-null int64
1 pclass
               891 non-null object
   sex
               714 non-null
                              float64
3
   age
   sibsp
                               int64
               891 non-null
   parch 891 non-null int64
fare 891 non-null float64
embarked 889 non-null object
class 891 non-null object
8 class
              891 non-null object
9 who
10 adult_male 891 non-null
                               bool
11 deck
                203 non-null object
12 embark_town 889 non-null object
13 alive 891 non-null
                                object
14 alone
                891 non-null
                                bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

Replacing outliers with Mean

```
In [17]: titanic_df = pd.read_csv("titanic.csv")

In [19]: m = np.mean(titanic_df['age'])
    print('mean:',m)
    for i in titanic_df['age']:
        if i<lower_bound or i>upper_bound:
            titanic_df['age'] = titanic_df['age'].replace(i,m)
```

mean: 29.081737106607342

Replacing outliers with median

```
In [25]: titanic_df = pd.read_csv("titanic.csv")

In [26]: q1 = titanic_df["age"].quantile(0.25)

# finding the 3rd quartile
q3 = titanic_df['age'].quantile(0.75)

# finding the iqr region
iqr = q3-q1

# finding upper and Lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)

In [27]: m = titanic_df['age'].median()
print(m)
for i in titanic_df['age']:
    if i<lower_bound or i>upper_bound:
        titanic_df['age'] = titanic_df['age'].replace(i,m)

28.0
```