#1. Import all the required Python Libraries.

In [1]:
```
import pandas as pd
```

#Data Wrangling I Perform the following operations using Python on any open source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas dataframe.
4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python. In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

#2. Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).

In [2]:
```
!pip install -q kaggle
```

In [4]:
```
from google.colab import files

files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Iris.csv to Iris.csv
```

Out [4]:

```
{'Iris.csv':
b'Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species\n1,5.1,3.5,1.4,0.2,Iris-
setosa\n2,4.9,3.0,1.4,0.2,Iris-setosa\n3,4.7,3.2,1.3,0.2,Iris-
setosa\n4,4.6,3.1,1.5,0.2,Iris-setosa\n5,5.0,3.6,1.4,0.2,Iris-
setosa\n6,5.4,3.9,1.7,0.4,Iris-setosa\n7,4.6,3.4,1.4,0.3,Iris-
setosa\n8,5.0,3.4,1.5,0.2,Iris-setosa\n9,4.4,2.9,1.4,0.2,Iris-
setosa\n10,4.9,3.1,1.5,0.1,Iris-setosa\n11,5.4,3.7,1.5,0.2,Iris-
setosa\n12,4.8,3.4,1.6,0.2,Iris-setosa\n13,4.8,3.0,1.4,0.1,Iris-
setosa\n14,4.3,3.0,1.1,0.1,Iris-setosa\n15,5.8,4.0,1.2,0.2,Iris-
setosa\n16,5.7,4.4,1.5,0.4,Iris-setosa\n17,5.4,3.9,1.3,0.4,Iris-
setosa\n18,5.1,3.5,1.4,0.3,Iris-setosa\n19,5.7,3.8,1.7,0.3,Iris-
setosa\n20,5.1,3.8,1.5,0.3,Iris-setosa\n21,5.4,3.4,1.7,0.2,Iris-
setosa\n22,5.1,3.7,1.5,0.4,Iris-setosa\n23,4.6,3.6,1.0,0.2,Iris-
setosa\n24,5.1,3.3,1.7,0.5,Iris-setosa\n25,4.8,3.4,1.9,0.2,Iris-
setosa\n26,5.0,3.0,1.6,0.2,Iris-setosa\n27,5.0,3.4,1.6,0.4,Iris-
setosa\n28,5.2,3.5,1.5,0.2,Iris-setosa\n29,5.2,3.4,1.4,0.2,Iris-
setosa\n30,4.7,3.2,1.6,0.2,Iris-setosa\n31,4.8,3.1,1.6,0.2,Iris-
setosa\n32,5.4,3.4,1.5,0.4,Iris-setosa\n33,5.2,4.1,1.5,0.1,Iris-
```

```
setosa\n34,5.5,4.2,1.4,0.2,Iris-setosa\n35,4.9,3.1,1.5,0.1,Iris-
setosa\n36,5.0,3.2,1.2,0.2,Iris-setosa\n37,5.5,3.5,1.3,0.2,Iris-
setosa\n38,4.9,3.1,1.5,0.1,Iris-setosa\n39,4.4,3.0,1.3,0.2,Iris-
setosa\n40,5.1,3.4,1.5,0.2,Iris-setosa\n41,5.0,3.5,1.3,0.3,Iris-
setosa\n42,4.5,2.3,1.3,0.3,Iris-setosa\n43,4.4,3.2,1.3,0.2,Iris-
setosa\n44,5.0,3.5,1.6,0.6,Iris-setosa\n45,5.1,3.8,1.9,0.4,Iris-
setosa\n46,4.8,3.0,1.4,0.3,Iris-setosa\n47,5.1,3.8,1.6,0.2,Iris-
setosa\n48,4.6,3.2,1.4,0.2,Iris-setosa\n49,5.3,3.7,1.5,0.2,Iris-
setosa\n50,5.0,3.3,1.4,0.2,Iris-setosa\n51,7.0,3.2,4.7,1.4,Iris-
versicolor\n52,6.4,3.2,4.5,1.5,Iris-versicolor\n53,6.9,3.1,4.9,1.5,Iris-
versicolor\n54,5.5,2.3,4.0,1.3,Iris-versicolor\n55,6.5,2.8,4.6,1.5,Iris-
versicolor\n56,5.7,2.8,4.5,1.3,Iris-versicolor\n57,6.3,3.3,4.7,1.6,Iris-
versicolor\n58,4.9,2.4,3.3,1.0,Iris-versicolor\n59,6.6,2.9,4.6,1.3,Iris-
versicolor\n60,5.2,2.7,3.9,1.4,Iris-versicolor\n61,5.0,2.0,3.5,1.0,Iris-
versicolor\n62,5.9,3.0,4.2,1.5,Iris-versicolor\n63,6.0,2.2,4.0,1.0,Iris-
versicolor\n64,6.1,2.9,4.7,1.4,Iris-versicolor\n65,5.6,2.9,3.6,1.3,Iris-
versicolor\n66,6.7,3.1,4.4,1.4,Iris-versicolor\n67,5.6,3.0,4.5,1.5,Iris-
versicolor\n68,5.8,2.7,4.1,1.0,Iris-versicolor\n69,6.2,2.2,4.5,1.5,Iris-
versicolor\n70,5.6,2.5,3.9,1.1,Iris-versicolor\n71,5.9,3.2,4.8,1.8,Iris-
versicolor\n72,6.1,2.8,4.0,1.3,Iris-versicolor\n73,6.3,2.5,4.9,1.5,Iris-
versicolor\n74,6.1,2.8,4.7,1.2,Iris-versicolor\n75,6.4,2.9,4.3,1.3,Iris-
versicolor\n76,6.6,3.0,4.4,1.4,Iris-versicolor\n77,6.8,2.8,4.8,1.4,Iris-
versicolor\n78,6.7,3.0,5.0,1.7,Iris-versicolor\n79,6.0,2.9,4.5,1.5,Iris-
versicolor\n80,5.7,2.6,3.5,1.0,Iris-versicolor\n81,5.5,2.4,3.8,1.1,Iris-
versicolor\n82,5.5,2.4,3.7,1.0,Iris-versicolor\n83,5.8,2.7,3.9,1.2,Iris-
versicolor\n84,6.0,2.7,5.1,1.6,Iris-versicolor\n85,5.4,3.0,4.5,1.5,Iris-
versicolor\n86,6.0,3.4,4.5,1.6,Iris-versicolor\n87,6.7,3.1,4.7,1.5,Iris-
versicolor\n88,6.3,2.3,4.4,1.3,Iris-versicolor\n89,5.6,3.0,4.1,1.3,Iris-
versicolor\n90,5.5,2.5,4.0,1.3,Iris-versicolor\n91,5.5,2.6,4.4,1.2,Iris-
versicolor\n92,6.1,3.0,4.6,1.4,Iris-versicolor\n93,5.8,2.6,4.0,1.2,Iris-
versicolor\n94,5.0,2.3,3.3,1.0,Iris-versicolor\n95,5.6,2.7,4.2,1.3,Iris-
versicolor\n96,5.7,3.0,4.2,1.2,Iris-versicolor\n97,5.7,2.9,4.2,1.3,Iris-
versicolor\n98,6.2,2.9,4.3,1.3,Iris-versicolor\n99,5.1,2.5,3.0,1.1,Iris-
versicolor\n100,5.7,2.8,4.1,1.3,Iris-versicolor\n101,6.3,3.3,6.0,2.5,Iris-
virginica\n102,5.8,2.7,5.1,1.9,Iris-virginica\n103,7.1,3.0,5.9,2.1,Iris-
virginica\n104,6.3,2.9,5.6,1.8,Iris-virginica\n105,6.5,3.0,5.8,2.2,Iris-
virginica\n106,7.6,3.0,6.6,2.1,Iris-virginica\n107,4.9,2.5,4.5,1.7,Iris-
virginica\n108,7.3,2.9,6.3,1.8,Iris-virginica\n109,6.7,2.5,5.8,1.8,Iris-
virginica\n110,7.2,3.6,6.1,2.5,Iris-virginica\n111,6.5,3.2,5.1,2.0,Iris-
virginica\n112,6.4,2.7,5.3,1.9,Iris-virginica\n113,6.8,3.0,5.5,2.1,Iris-
virginica\n114,5.7,2.5,5.0,2.0,Iris-virginica\n115,5.8,2.8,5.1,2.4,Iris-
virginica\n116,6.4,3.2,5.3,2.3,Iris-virginica\n117,6.5,3.0,5.5,1.8,Iris-
virginica\n118,7.7,3.8,6.7,2.2,Iris-virginica\n119,7.7,2.6,6.9,2.3,Iris-
virginica\n120,6.0,2.2,5.0,1.5,Iris-virginica\n121,6.9,3.2,5.7,2.3,Iris-
virginica\n122,5.6,2.8,4.9,2.0,Iris-virginica\n123,7.7,2.8,6.7,2.0,Iris-
virginica\n124,6.3,2.7,4.9,1.8,Iris-virginica\n125,6.7,3.3,5.7,2.1,Iris-
virginica\n126,7.2,3.2,6.0,1.8,Iris-virginica\n127,6.2,2.8,4.8,1.8,Iris-
virginica\n128,6.1,3.0,4.9,1.8,Iris-virginica\n129,6.4,2.8,5.6,2.1,Iris-
virginica\n130,7.2,3.0,5.8,1.6,Iris-virginica\n131,7.4,2.8,6.1,1.9,Iris-
virginica\n132,7.9,3.8,6.4,2.0,Iris-virginica\n133,6.4,2.8,5.6,2.2,Iris-
virginica\n134,6.3,2.8,5.1,1.5,Iris-virginica\n135,6.1,2.6,5.6,1.4,Iris-
virginica\n136,7.7,3.0,6.1,2.3,Iris-virginica\n137,6.3,3.4,5.6,2.4,Iris-
virginica\n138,6.4,3.1,5.5,1.8,Iris-virginica\n139,6.0,3.0,4.8,1.8,Iris-
virginica\n140,6.9,3.1,5.4,2.1,Iris-virginica\n141,6.7,3.1,5.6,2.4,Iris-
virginica\n142,6.9,3.1,5.1,2.3,Iris-virginica\n143,5.8,2.7,5.1,1.9,Iris-
virginica\n144,6.8,3.2,5.9,2.3,Iris-virginica\n145,6.7,3.3,5.7,2.5,Iris-
virginica\n146,6.7,3.0,5.2,2.3,Iris-virginica\n147,6.3,2.5,5.0,1.9,Iris-
virginica\n148,6.5,3.0,5.2,2.0,Iris-virginica\n149,6.2,3.4,5.4,2.3,Iris-
virginica\n150,5.9,3.0,5.1,1.8,Iris-virginica\n'}
```


#3. Load the Dataset into pandas dataframe.

In [5]:

```
iris = pd.read_csv("/content/Iris.csv")
iris
```

Out [5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

4. Data Preprocessing:

check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.

Print a concise summary of a DataFrame.

In [6]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Return the first n rows.

In [7]:

```
iris.head(10)
```

Out [7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

Return the last n rows.

In [8]:
```
iris.tail(15)
```

Out [8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
135	136	7.7	3.0	6.1	2.3	Iris-virginica
136	137	6.3	3.4	5.6	2.4	Iris-virginica
137	138	6.4	3.1	5.5	1.8	Iris-virginica
138	139	6.0	3.0	4.8	1.8	Iris-virginica
139	140	6.9	3.1	5.4	2.1	Iris-virginica
140	141	6.7	3.1	5.6	2.4	Iris-virginica
141	142	6.9	3.1	5.1	2.3	Iris-virginica
142	143	5.8	2.7	5.1	1.9	Iris-virginica
143	144	6.8	3.2	5.9	2.3	Iris-virginica
144	145	6.7	3.3	5.7	2.5	Iris-virginica
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

Generate descriptive statistics.

In [9]:
```
iris.describe(include = "all")
```

Out [9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

Return a tuple representing the dimensionality of the DataFrame.

In [10]:
```
iris.shape
```

Out [10]:
```
(150, 6)
```

The column labels of the DataFrame.

In [11]:
```
iris.columns
```

Out [11]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [12]:
```
iris.Species
```

Out [12]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

Gives the content of a coloum

In [16]:
```
iris["Id"]
```

Out [16]:
```
0        1
1        2
2        3
3        4
4        5
       ...
145    146
146    147
147    148
148    149
149    150
Name: Id, Length: 150, dtype: int64
```

In [17]:
```
iris[0:3]
```

Out [17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

Access a group of rows and columns by label(s) or a boolean array.

In [18]:
```
iris.loc[0:2]
```

Out [18]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

In [21]:
```
iris.loc[0:2,"Id":"PetalWidthCm"]
```

Out [21]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2

Purely integer-location based indexing for selection by position.

In [23]:
```
iris.iloc[1:6]
```

Out [23]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
5	6	5.4	3.9	1.7	0.4	Iris-setosa

In [24]:
```
iris.iloc[1:5,1:5]
```

Out [24]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Detect missing values.

In [25]:
```
iris.isnull()
```

Out [25]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

In [26]:
```
iris.isna()
```

Out [26]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

Return whether any element is True, potentially over an axis.

In [27]:
```
iris.isna().any()
```

Out [27]:
```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

Return the sum of the values over the requested axis.

In [28]:
```
iris.isnull().sum()
```

Out [28]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

count of missing values of a speci�c column.

In [29]:
```
iris.SepalLengthCm.isnull().sum()
```

Out [29]: 0

#5. Data Formatting and Data Normalization:

Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.

##Data Formatting

Return the dtypes in the DataFrame.

In [32]:
```
iris.dtypes
```

Out [32]:
```
Id                 int64
SepalLengthCm      int64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

Cast a pandas object to a specified dtype dtype.

In [30]:
```
iris.SepalLengthCm  = iris.SepalLengthCm.astype("int")
```

In [33]:
```
iris.dtypes
```

Out [33]:
```
Id                 int64
SepalLengthCm      int64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

##Data Normalization

In [34]:
```
from sklearn import preprocessing # step1 :Import pandas and sklearn lib
```

In [35]:
```
iris.head() #step2: Load the iris dataset in dataframe object df
```

Out [35]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5	3.5	1.4	0.2	Iris-setosa
1	2	4	3.0	1.4	0.2	Iris-setosa
2	3	4	3.2	1.3	0.2	Iris-setosa
3	4	4	3.1	1.5	0.2	Iris-setosa
4	5	5	3.6	1.4	0.2	Iris-setosa

Transform features by scaling each feature to a given range.

In [36]:
```
min_max_scaler = preprocessing.MinMaxScaler() #min-max scalar
```

In [38]:
```
x = iris.iloc[:,:4]
x
```

Out [38]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5	3.5	1.4
1	2	4	3.0	1.4
2	3	4	3.2	1.3
3	4	4	3.1	1.5
4	5	5	3.6	1.4
...	...	...	...	...
145	146	6	3.0	5.2
146	147	6	2.5	5.0
147	148	6	3.0	5.2
148	149	6	3.4	5.4
149	150	5	3.0	5.1

150 rows × 4 columns

Fit(Compute the minimum and maximum to be used for later scaling) to data, then transform(Scale features of X according to feature_range.) it.

In [39]:

```
x_scaled = min_max_scaler.fit_transform(x) # Create an object to transfo
```

In [41]:

```
df_normalized = pd.DataFrame(x_scaled) #normalized data
df_normalized
```

Out [41]:

	0	1	2	3
0	0.000000	0.333333	0.625000	0.067797
1	0.006711	0.000000	0.416667	0.067797
2	0.013423	0.000000	0.500000	0.050847
3	0.020134	0.000000	0.458333	0.084746
4	0.026846	0.333333	0.666667	0.067797
...	...	...	...	...
145	0.973154	0.666667	0.416667	0.711864
146	0.979866	0.666667	0.208333	0.677966
147	0.986577	0.666667	0.416667	0.711864
148	0.993289	0.666667	0.583333	0.745763
149	1.000000	0.333333	0.416667	0.694915

150 rows × 4 columns

In [42]:
```
iris
```

Out [42]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5	3.5	1.4	0.2	Iris-setosa
1	2	4	3.0	1.4	0.2	Iris-setosa
2	3	4	3.2	1.3	0.2	Iris-setosa
3	4	4	3.1	1.5	0.2	Iris-setosa
4	5	5	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6	3.0	5.2	2.3	Iris-virginica
146	147	6	2.5	5.0	1.9	Iris-virginica
147	148	6	3.0	5.2	2.0	Iris-virginica
148	149	6	3.4	5.4	2.3	Iris-virginica
149	150	5	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

#6. Turn categorical variables into quantitative variables in Python.

There are many ways to convert categorical data into numerical data. Here the three most used methods are discussed.

##i. Label Encoding:

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. It is an important preprocessing step for the structured dataset in supervised learning.

In [43]:
```
from sklearn import preprocessing
```

Return unique values of Series object.

In [44]:
```
iris['Species'].unique()
```

Out [44]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Encode target labels with value between 0 and n_classes-1.

In [45]:
```
label_encoder = preprocessing.LabelEncoder()
```

Fit label encoder and return encoded labels.

In [46]:
```
iris['Species']= label_encoder.fit_transform(iris['Species'])
```

In [47]:
```
iris['Species'].unique()
```

Out [47]:
```
array([0, 1, 2])
```

In [48]:
```
iris
```

Out [48]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5	3.5	1.4	0.2	0
1	2	4	3.0	1.4	0.2	0
2	3	4	3.2	1.3	0.2	0
3	4	4	3.1	1.5	0.2	0
4	5	5	3.6	1.4	0.2	0
...	...	...	...	...	...	...
145	146	6	3.0	5.2	2.3	2
146	147	6	2.5	5.0	1.9	2
147	148	6	3.0	5.2	2.0	2
148	149	6	3.4	5.4	2.3	2
149	150	5	3.0	5.1	1.8	2

150 rows × 6 columns

#Conclusion

In this way we have explored the functions of the python library for Data Preprocessing, Data Wrangling Techniques and How to Handle missing values on Iris Dataset.

In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.