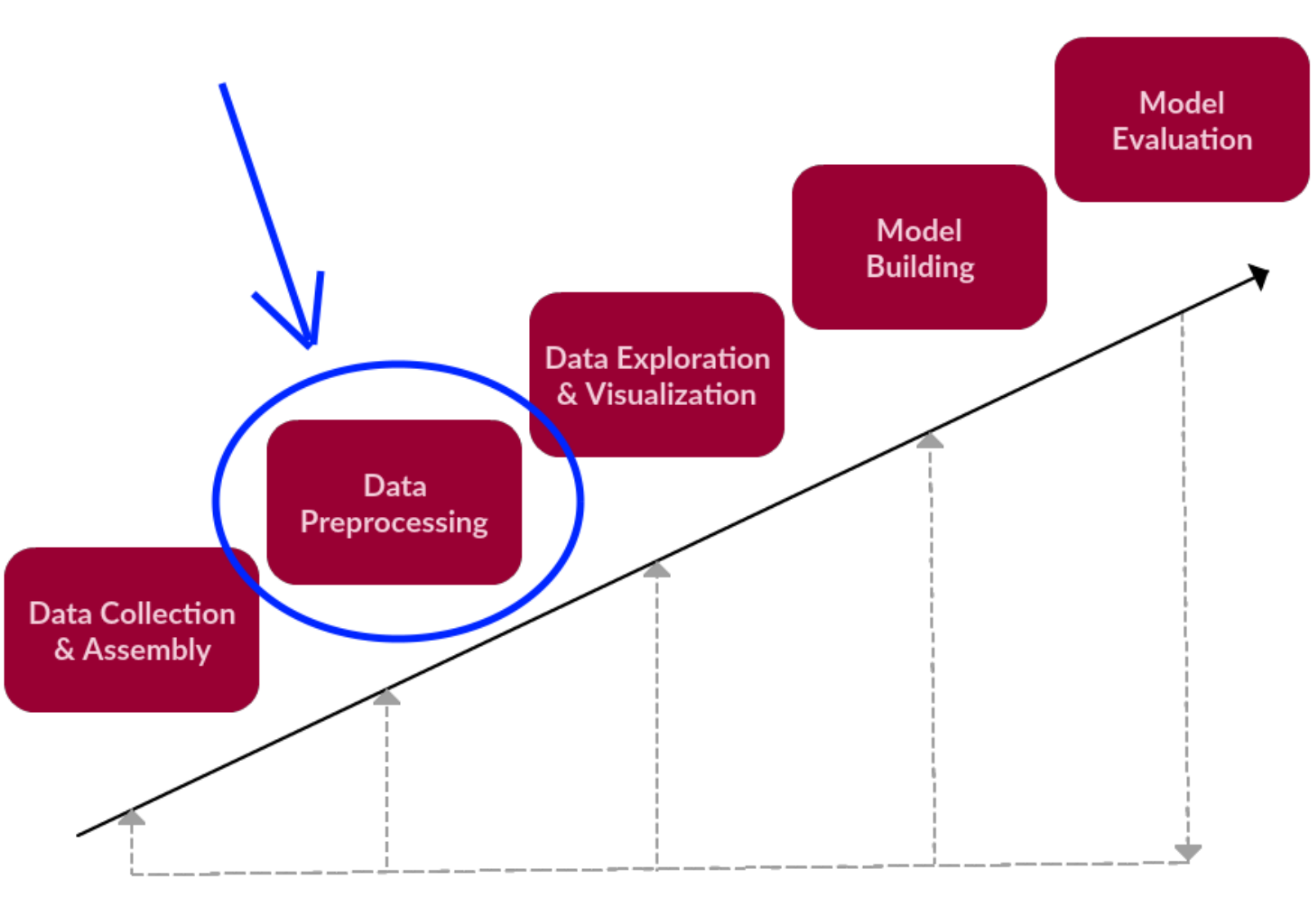
Data Preprocessing - Data Analysis

Data Preprocessing

It is used for maintaining the **Quality** of the data. It includes important factors like -

- Selecting the valid data variables
- Data editing is important in some aspects
- Maintaining uniformity in data values
- Manipulation of the data for achieving the above factors (Data Wrangling)



Credits - Image from Internet

Classroom (Career) Management

Imagine you are the new teacher freshly appointed to manage the classroom.

You want to know how many students are good in -

```
1. sports
2. academics
3. creative work
4. marketing
```

Based on the students data, you have to conclude **who** can do well in **what**.

Dataset description

You are given the **data of the students** that included the following variables -

- Age
- Gender
- Address
- Father's occupation
- Mother's occupation
- Place of birth
- Height - ft
- Weight - kg
- Prev sports performance
- Prev academics performance
- Voluntary experience
- Extra co-curricular activities
- Arts and Design

Sports

Scenario 1

Considering the variables that are direclty related -

- Height
- Weight
- Prev sports performance

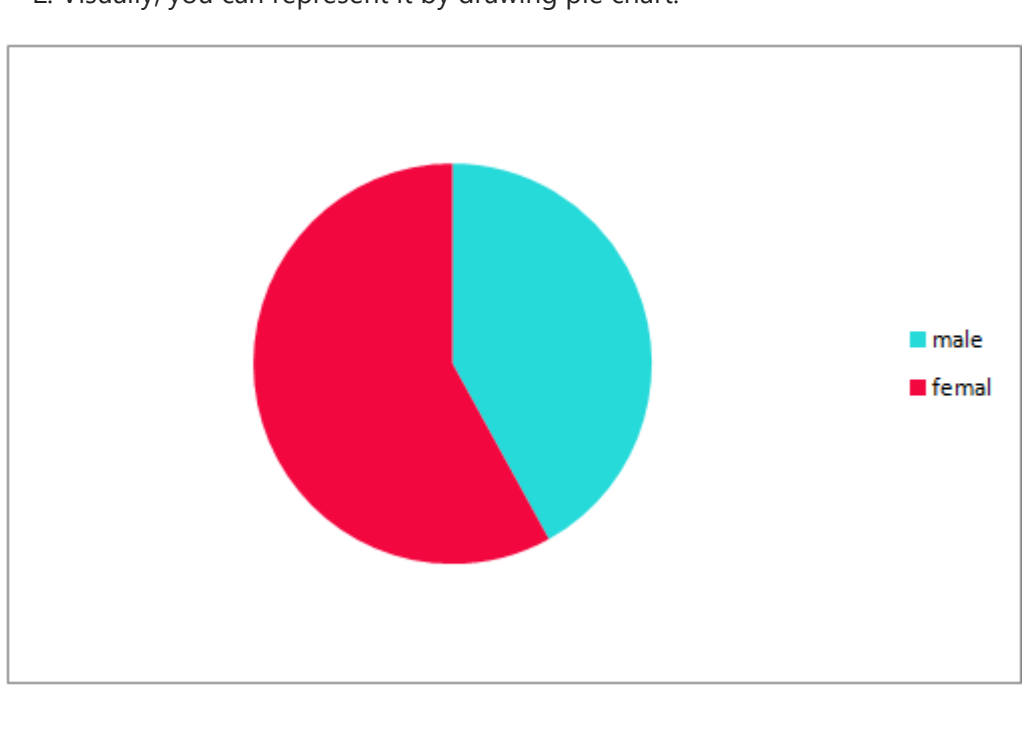
Based on this, you can only get the information of a student irrespective of **gender**.

Scenario 2

Considering the other important factors like **gender** in order to categorize as per **Male related sports** and **Female related sports**.

- Gender
- Height
- Weight
- Prev sports performance

1. Based on this, you can categorize the performance of students in sports by **Male** and **Female**.
2. Visually, you can represent it by drawing pie chart.



Credits - Image from Internet

Scenario 3

If you want to do further research on how good the person is performing in other areas, you can do so by considering -

- Gender
- Height
- Weight
- Prev sports performance
- Voluntary experience
- Extra co-curricular activities
- Prev academics performace (may be or may not be)

1. With this, you conclude the overall students performance on sports.
2. Thus, you can give students a proper career guidance for their betterment.

Academics

Scenario 1

Considering the variables that are directly related -

- Prev academics peroformance

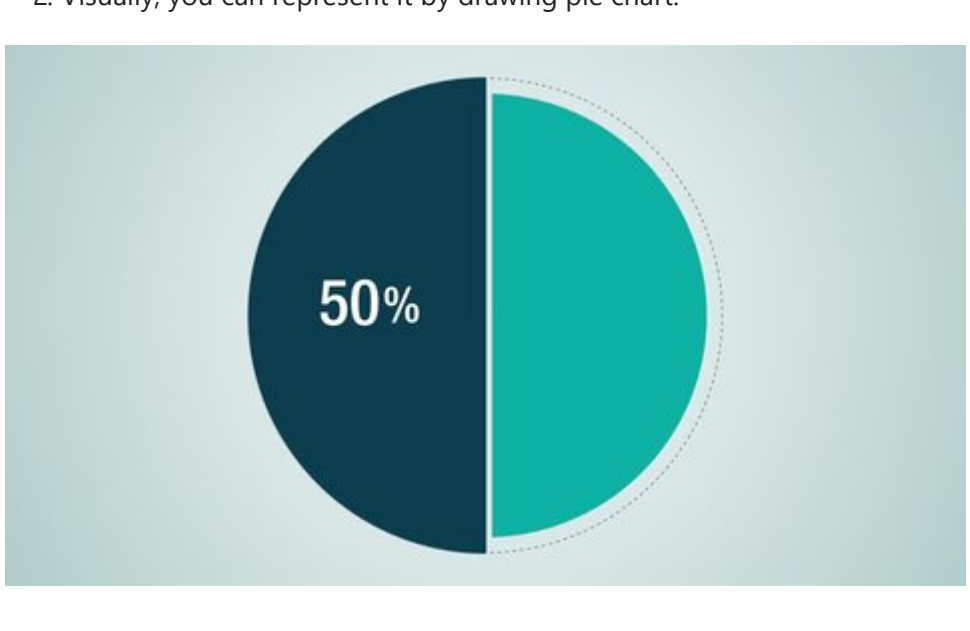
Based on this, you can only get the information of student irrespective of **gender**.

Scenario 2

Considering the other important factors like **gender** to categorize **Male** and **Female** separately.

- Gender
- Prev academics performance

1. Based on this, you can categorize the performance of students in academics by **Male** and **Female**.
2. Visually, you can represent it by drawing pie chart.



Credits - Image from Internet

Scenario 3

If you want to further research on why a particular student is **lagging behind** or **excelling ahead**, you can do so by considering -

- Address
- Father's occupation
- Mother's occupation
- Prev academics performance
- Gender (for categorizing in terms of gender)

and later on, you can decide whether to change your teaching methodology or not.

Note -

- Data Analyst should be wise enough to select the important data variables.
- This helps to get proper insights pertaining to the problem statement that he/she is assigned to do.

In [ ]:

How do we convert a **" feeling "** into a number?

- We can measure a **" feeling "** into a number through a **" scale range "**
- If the scale is **1** to **4**, then we can term -
  - 1 → Not Satisfied
  - 2 → Slightly Satisfied
  - 3 → Satisfied
  - 4 → Highly Satisfied

In [ ]:

Let's make our hands dirty

Source → https://bit.ly/3g6AEPj

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

In [2]:
```
data_source = 'https://bit.ly/3g6AEPj'
# df
df = pd.read_csv(data_source)
```

In [3]:
```
# type
type(df)
```

Out[3]: `pandas.core.frame.DataFrame`

In [4]:
```
# head
df.head()
```

Out[4]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

Check the length of the `df`

In [5]:
```
# shape
df.shape
```

Out[5]: `(200, 2)`

Data Preprocessing

Check for `NaN`

In [6]:
```
# isnull().any()
df.isnull().any()
```

Out[6]:
```
Height(Inches)    True
Weight(Pounds)    True
dtype: bool
```

In [7]:
```
# isnull().sum()
df.isnull().sum()
```

Out[7]:
```
Height(Inches)    2
Weight(Pounds)    1
dtype: int64
```

In [8]:
```
# list of columns
list(df.columns)
```

Out[8]: `['Height(Inches)', 'Weight(Pounds)']`

Things to read

- What is dictionary in Python?
  - Keys and Values pairing. Refer to this link.
- What is a function?
- How to define functions?
- How to call functions?
- Types of functions

In [9]:
```
for col in df.columns:
    print(col)
```

```
Height(Inches)
Weight(Pounds)
```

Hey Python, take help of `numpy` to locate the `NaN` values for each column in dataframe `df` and save it as a dictionary.

In [10]:
```
def get_nan_indices(dframe):
    """
    dframe → pandas data frame object
    returns `nan_places` a dictionary of column names and the `nan_indices`
    """
    nan_places = {}

    for col in dframe.columns:
        indices = list(np.where(dframe[col].isnull())[0])
        nan_places[col] = indices

    return nan_places
```

In [11]:
```
# function call
get_nan_indices(dframe=df)
```

Out[11]: `{'Height(Inches)': [10, 32], 'Weight(Pounds)': [19]}`

1. In the column `Height(Inches)`, there are two `NaN` values at indices `10` and `32`.
2. In the column `Weight(Pounds)`, there is one `NaN` value at index `19`.

What can we do for those?

- Remove the entire row which ever column has a `NaN`.

For this, we will remove the rows which ever column has `NaN`. In total, there are 3 rows that need to be removed.

Remove 3 rows

- axis (0) → row
- axis (1) → column

In [12]:
```
# df_1 → removing by index
df_1 = df.drop(index=[10, 19, 32], axis=0)
```

In [13]:
```
# df_1 → shape
df_1.shape
```

Out[13]: `(197, 2)`

In [14]:
```
# df dropna - pdf
pdf = df.dropna(axis=0)
```

Check the length of `pdf`

In [15]:
```
# shape
pdf.shape
```

Out[15]: `(197, 2)`

Since the index of the data frame is not in order, we need to reindex the index values to get the perfect order.

In [16]:
```
# head(12)
pdf.head(12)
```

Out[16]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30
5	68.70	123.30
6	69.80	141.49
7	70.01	136.46
8	67.90	112.37
9	66.78	120.67
11	67.62	114.14
12	68.30	125.61

Reset the index

In [17]:
```
# rdf
# reset with drop
rdf = pdf.reset_index(drop=True)
```

In [18]:
```
# head
rdf.head(12)
```

Out[18]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30
5	68.70	123.30
6	69.80	141.49
7	70.01	136.46
8	67.90	112.37
9	66.78	120.67
10	67.62	114.14
11	68.30	125.61

In [19]:
```
# shape
rdf.shape
```

Out[19]: `(197, 2)`

Check if `Height(Inches)` < 40

In [20]:
```
# inch_thresh
inch_thresh = 40
```

In [21]:
```
# filter with <
rdf[rdf['Height(Inches)'] < inch_thresh]
```

Out[21]:

	Height(Inches)	Weight(Pounds)
68	30.84	134.02
93	36.29	120.03

Remove the rows where `Height(Inches)` < 40

- In the above case, we can see two values where height is less than 40.
- We remove by specifying the index values in `drop()` method.

In [22]:
```
# drop by index
rdf = rdf.drop(index=[68, 93], axis=0)
```

In [23]:
```
# shape
rdf.shape
```

Out[23]: `(195, 2)`

Reset the index

In [24]:
```
# hw_df
# drop = True
hw_df = rdf.reset_index(drop=True)
```

In [25]:
```
# shape
hw_df.shape
```

Out[25]: `(195, 2)`

Since the index of the data frame is not in order, we need to reindex the index values to get the perfect order.

Categorize the data

- Refer to → https://pandas.pydata.org/docs/reference/api/pandas.cut.html

In [26]:
```
# head
hw_df.head()
```

Out[26]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

In [27]:
```
# height_cat
# weight_cat

height_cat = pd.cut(x=hw_df['Height(Inches)'], bins=3, labels=['short', 'average', 'tall'])
weight_cat = pd.cut(x=hw_df['Weight(Pounds)'], bins=3, labels=['under weight', 'normal weight', 'obesity'])
```

Make a new column `height_cat` and `weight_cat` in the dataframe - `hw_df`

In [28]:
```
# make new columns
hw_df['height_cat'] = height_cat
hw_df['weight_cat'] = weight_cat
```

In [29]:
```
# head
hw_df.head()
```

Out[29]:

	Height(Inches)	Weight(Pounds)	height_cat	weight_cat
0	65.78	112.99	short	under weight
1	71.52	136.49	tall	normal weight
2	69.40	153.03	average	obesity
3	68.22	142.34	average	obesity
4	67.79	144.30	average	obesity

Plotting the pie chart to show

- how many are short
- how many are tall
- ...

Get `value_counts()` from `height_cat` variable

In [30]:
```
hw_df['height_cat'].value_counts()
```

Out[30]:
```
average    118
short       60
tall        17
Name: height_cat, dtype: int64
```

Get `value_counts()` from `weight_cat` variable

In [31]:
```
hw_df['weight_cat'].value_counts()
```

Out[31]:
```
normal weight    114
under weight      44
obesity           37
Name: weight_cat, dtype: int64
```

In [32]:
```
# hdf_pie → to_frame()
# wdf_pie → to_frame()

hdf_pie = hw_df['height_cat'].value_counts().to_frame()
wdf_pie = hw_df['weight_cat'].value_counts().to_frame()
```

In [33]:
```
# display hdf_pie
hdf_pie
```

Out[33]:

	height_cat
average	118
short	60
tall	17

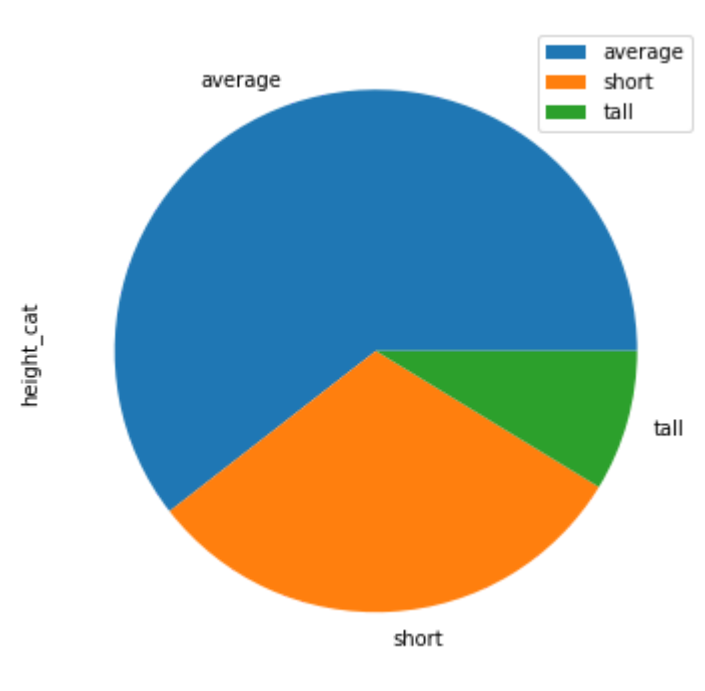
In [34]:
```
# display wdf_pie
wdf_pie
```

Out[34]:

	weight_cat
normal weight	114
under weight	44
obesity	37

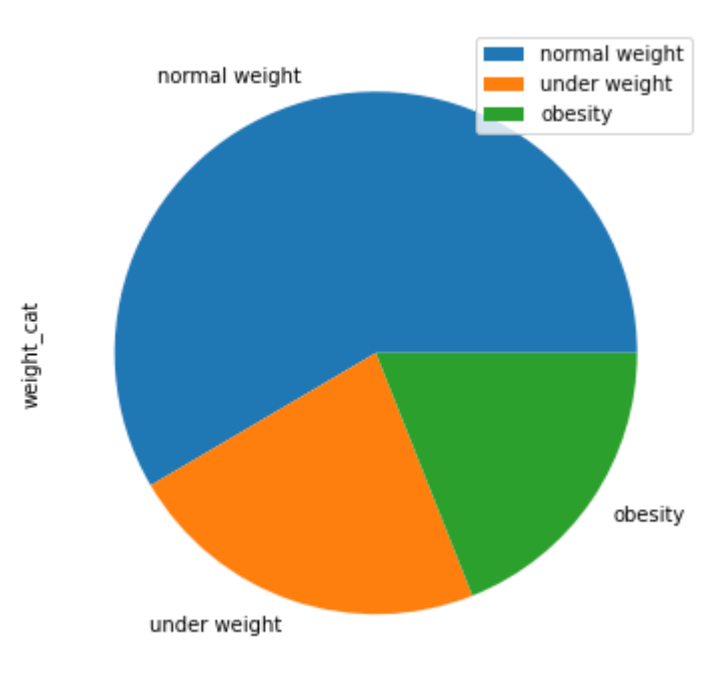
In [35]:
```
# plot pie of hdf_pie with size (width=10, height=6)
hdf_pie.plot(figsize=(10, 6), kind='pie', subplots=True)
```

Out[35]: `array([<AxesSubplot:ylabel='height_cat'>], dtype=object)`



In [36]:
```
# plot pie of wdf_pie with size (width=10, height=6)
wdf_pie.plot(figsize=(10, 6), kind='pie', subplots=True)
```

Out[36]: `array([<AxesSubplot:ylabel='weight_cat'>], dtype=object)`



What did we learn?

- Data Preprocessing
- Real life scenario Example
- Classroom Management problem
  - sports
  - academics
  - creative work
  - marketing
- Getting hands dirty by writing code
- Plotting the pie chart for showing categorization