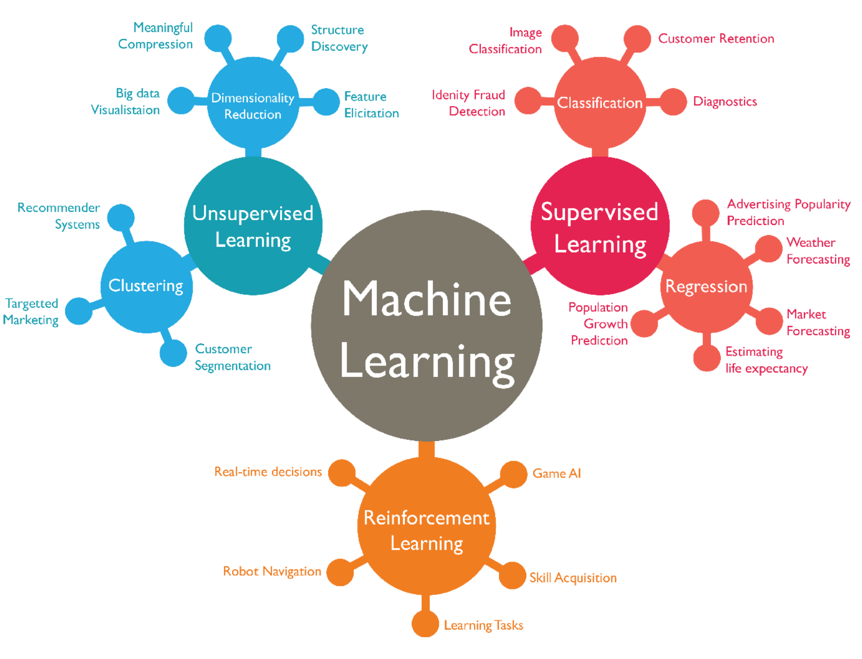
Machine Learning

- ML is a technique followed to make a computer learn from the previous experience and make an assumption for the future outcome.
- It can learn and adapt to the new data without any human intervention.
- It needs prior training so that it can be tested to the new data.

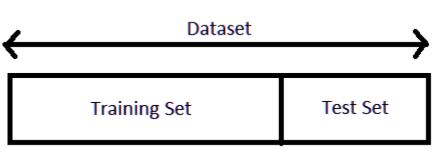


Credits - Image from Internet (www.favouriteblog.com)

In [ ]:

ML Dataset

In machine learning, we divide the dataset into two.



- **Training Data** - Here we train the machine learning model by showing both inputs and outputs.
- **Testing Data** - Here we test the model where inputs (new data) are not mapped with outputs. We will check the model performance in terms of accuracy.

Credits - Image from Internet

In [ ]:

Supervised Learning

The computer is presented with both example inputs and their respective outputs. The algorithm learns a general rule to map the inputs to the outputs.

Training stage

We show the model a set of inputs along with the respective outputs. The task of the model is to learn by mapping the inputs and outputs.

The model is trained from the dataset that is fed. It will completely learn from it.

Testing stage

We show the model a set of new inputs without the respective outputs. The aim of the model is to predict based on the learning it had undergone.

The model predicts the category based on the previous training or learning.

Images by Author

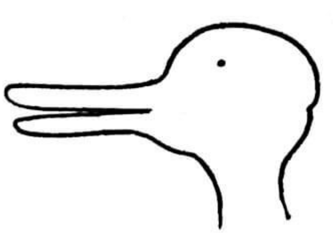
Note

- Algorithms learn from data.
- They find -
  - relationships
  - develop understanding
  - make decisions
  - evaluate their confidence from the training data they are given.
- The better the training data is, the better the model performs.

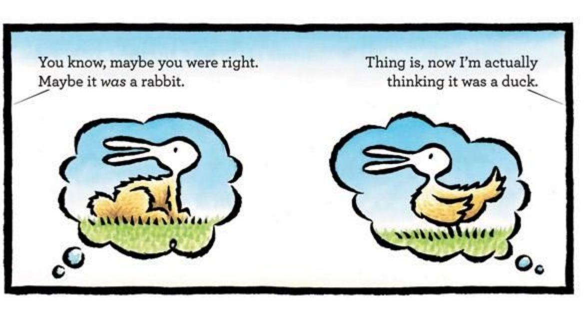
Exception case for the above example

1. Suppose I have trained my model to identify/separate `duck` and `rabbit` images from the large dataset.

1. Now, I need to test my model with the new data.
   - New image



   - Is it duck or rabbit?



2. What can you say about this?

Credits - Images from Internet

In [ ]:

Regression

- Regression is a process of predicting the dependant variable based on the independant variable.
- Dependant variable is always considered as Y.
- Independant variable is always considered as x.
- For doing regression analysis, there needs to be a strong relationship betweem x and Y.

Example - Predicting the expenses of an employee based on his/her income.

- Here,
  - expenses are dependant variable $\rightarrow Y$
  - income is independant variable $\rightarrow x$

How can be predict Y from x?

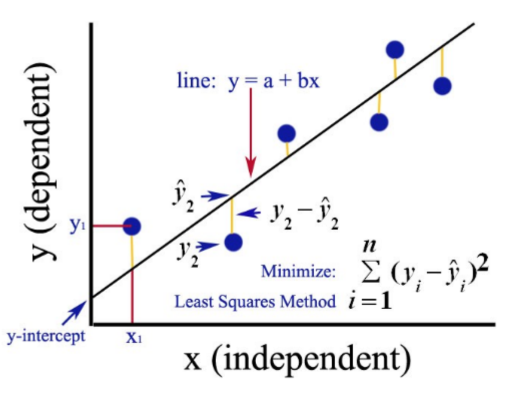
To predict Y from x, we follow a mathematical model -

$$Y = bx + a$$

- Here,
  - $b \rightarrow$ Co-efficient parameter
  - $a \rightarrow$ Bias parameter
- Other terms
  - $\hat{Y}$ (Y hat) $\rightarrow$ Predicted Y value
  - $Y \rightarrow$ Actual value

What is Error?

- The distance between $\hat{Y}$ and Y is called Error.
- Our aim while building the Regression model is to minimize the Error.



Credits - Image from Internet

In [ ]:

In [ ]:

Let's predict something ... !

`import` packages

In [1]:
```python
import warnings
warnings.filterwarnings("ignore")
```

To ignore warnings that arise while executing the library method

In [2]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

If you do not have `seaborn`

```
pip install seaborn --user
```

Data exploration

In [3]:
```python
# read data
df = pd.read_csv('brain_body_weight.csv')
```

In [4]:
```python
# shape
df.shape
```

Out[4]: (62, 3)

In [5]:
```python
# head
df.head()
```

Out[5]:

	Index	Brain Weight	Body Weight
0	1	3.385	44.5
1	2	0.480	15.5
2	3	1.350	8.1
3	4	465.000	423.0
4	5	36.330	119.5

In [6]:
```python
# remove Index
df = df.drop(columns=['Index'], axis=1)
```

In [7]:
```python
# shape
df.shape
```

Out[7]: (62, 2)

In [8]:
```python
# head
df.head()
```

Out[8]:

	Brain Weight	Body Weight
0	3.385	44.5
1	0.480	15.5
2	1.350	8.1
3	465.000	423.0
4	36.330	119.5

- Dependant variable $(Y) \rightarrow$ Brain Weight
- Independant variable $(x) \rightarrow$ Body Weight

In [9]:
```python
# Create Y and x (array)
Y = np.array(df['Brain Weight'])
x = np.array(df['Body Weight'])
```

How x is related to Y?

In [10]:
```python
# correlation
bb_corr = df.corr()
```

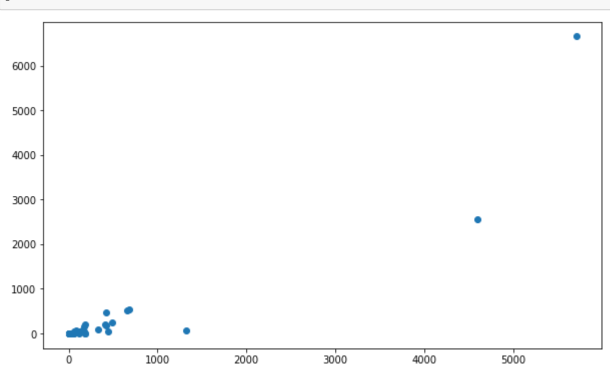
In [11]:
```python
bb_corr
```

Out[11]:

	Brain Weight	Body Weight
Brain Weight	1.000000	0.934164
Body Weight	0.934164	1.000000

From above, we can say `Brain Weight` and `Body Weight` are highly correlated.

In [12]:
```python
# scatter plot (10, 6)
plt.figure(figsize=(10, 6))
plt.scatter(x, Y)
plt.show()
```



There are outliers in our data.

Outlier detection function

In [13]:
```python
def calculate_zscore(data_values):
    # mean
    mean_vals = np.mean(data_values)
    # standard deviation
    std_dev = np.std(data_values)
    # applying the formula for all the values
    zscore = np.array([(i - mean_vals)/std_dev for i in data_values])
    return zscore
```

Model Implementation

In [14]:
```python
# b, a - polyfit
b, a = np.polyfit(x, Y, 1) # Gradient descent
```

In [15]:
```python
# b
b
```

Out[15]: 0.9029129477287059

In [16]:
```python
# a
a
```

Out[16]: -56.855545428596535

In [17]:
```python
x
```

Out[17]:
```
array([4.450e+01, 1.550e+01, 8.100e+00, 4.230e+02, 1.195e+02, 1.150e+02,
       9.820e+01, 5.500e+00, 5.800e+01, 6.400e+00, 4.000e+00, 5.700e+00,
       6.600e+00, 1.400e-01, 1.000e+00, 1.080e+01, 1.230e+01, 6.300e+00,
       4.603e+03, 3.000e-01, 4.190e+02, 6.550e+02, 3.500e+00, 1.150e+02,
       2.560e+01, 5.000e+00, 1.750e+01, 6.800e+02, 4.060e+02, 3.250e+02,
       1.230e+01, 1.320e+03, 5.712e+03, 3.900e+00, 1.790e+02, 5.600e+01,
       1.700e+01, 1.000e+00, 4.000e-01, 2.500e-01, 1.250e+01, 4.900e+02,
       1.210e+01, 1.750e+02, 1.570e+02, 4.400e+02, 1.795e+02, 2.400e+00,
       8.100e+01, 2.100e+01, 3.920e+01, 1.900e+00, 1.200e+00, 3.000e+00,
       3.300e-01, 1.800e+02, 2.500e+01, 1.690e+02, 2.600e+00, 1.140e+01,
       2.500e+00, 5.040e+01])
```

Broadcasting

In [18]:
```python
s = np.array([1, 2, 3, 5])
v = 3
s*v # [3, 6, 9, 15]
```

Out[18]: array([3, 6, 9, 15])

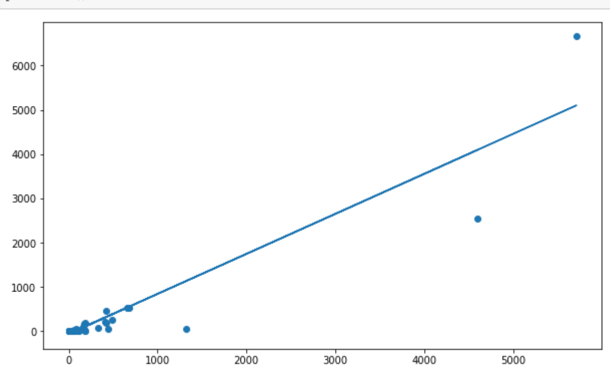
In [19]:
```python
# y_preds = bx + a
y_preds = b*x + a
```

In [20]:
```python
# y_preds
y_preds
```

Out[20]:
```
array([-1.66759193e+01, -4.28603947e+01, -4.95419506e+01,  3.25076631e+02,
        5.10425518e+01,  4.69794436e+01,  3.18105060e+01, -5.18895242e+01,
       -4.48659446e+00, -5.10769026e+01, -5.32438936e+01, -5.17089416e+01,
       -5.08963200e+01, -5.67291376e+01, -5.59526325e+01, -4.71040856e+01,
       -4.57497162e+01, -5.11671939e+01,  4.09925275e+03, -5.65846715e+01,
        3.21464980e+02,  5.34552435e+02, -5.36953501e+01,  4.69794436e+01,
       -3.37409740e+01, -5.23409807e+01, -4.10545688e+01,  5.57125259e+02,
        3.09727111e+02,  2.36591163e+02, -4.57497162e+01,  1.13498955e+03,
        5.10058321e+03, -5.33341849e+01,  1.04765872e+02, -6.29242036e+00,
       -4.15060253e+01, -5.59526325e+01, -5.64943802e+01, -5.66298172e+01,
       -4.55691336e+01,  3.85571799e+02, -4.59302988e+01,  1.01154220e+02,
        8.49017874e+01,  3.40426152e+02,  1.05217329e+02, -5.46885544e+01,
        1.62804033e+01, -3.78943735e+01, -2.14613579e+01, -5.51400108e+01,
       -5.57720499e+01, -5.41468066e+01, -5.65575842e+01,  1.05668785e+02,
       -3.42827217e+01,  9.57367427e+01, -5.45079718e+01, -4.65623378e+01,
       -5.45982631e+01, -1.13487329e+01])
```

Best fit line

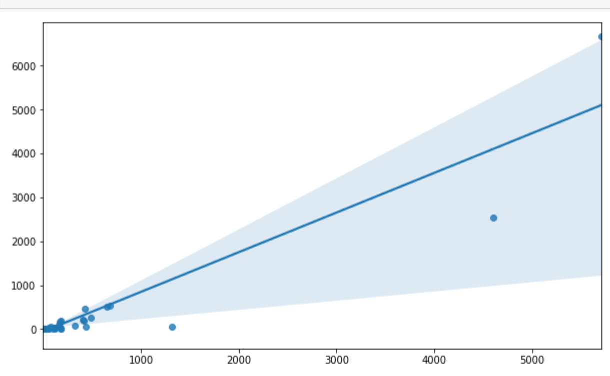
In [21]:
```python
# scatter and line
plt.figure(figsize=(10, 6))
plt.scatter(x ,Y)
plt.plot(x, y_preds)
plt.show()
```



In [ ]:

Seaborn plot with `confidence` level

In [22]:
```python
# confidence level
plt.figure(figsize=(10, 6))
sns.regplot(x, Y)
plt.show()
```



In [ ]:

Plotly plot for interactivity

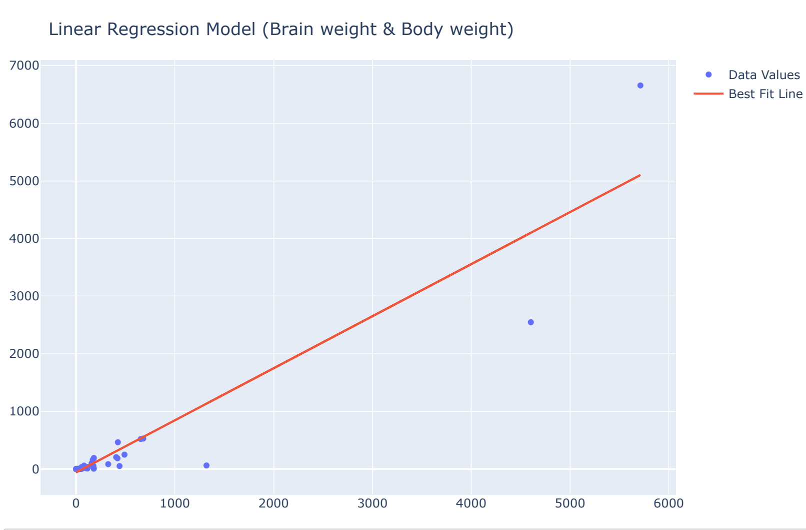
In [23]:
```python
import plotly.graph_objects as go
```

In [24]:
```python
trace1 = go.Scatter(
    x=x,
    y=Y,
    mode='markers',
    name='Data Values'
)

trace2 = go.Scatter(
    x=x,
    y=y_preds,
    mode='lines',
    name='Best Fit Line'
)

layout = go.Layout(
    title='Linear Regression Model (Brain weight & Body weight)',
    height=500,
    width=800,
    margin=dict(l=0, b=10, t=50, r=0)
)

fig = go.Figure(data=[trace1, trace2], layout=layout)
fig.show()
```



In [ ]:

What about Multiple Linear Regression model?

Dependant variable is depends on mutiple Independant variables.

Model -

$$Y = b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n + a$$

- $x_1, x_2, x_3, \ldots, x_n$ are independant variables.
- $b_1, b_2, b_3, \ldots, b_n$ are coefficient variables.
- a is bias variable.

The main aim of any machine learning model is to minimize the error.

In [ ]: