X = -1, 0, 1, 2, 3, 4

Y = -3, -1, 1, 3, 5, 7

Formula here : 2X - 1

```
pip install --upgrade keras
```

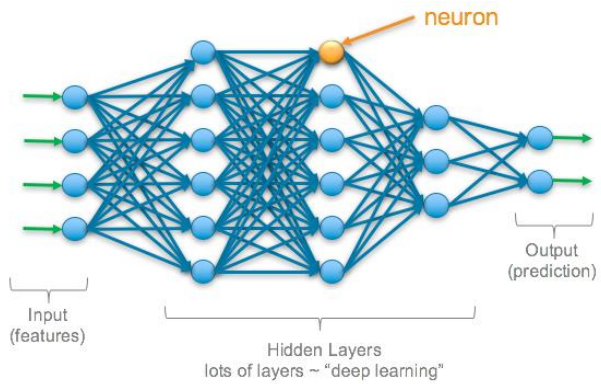
```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

model.predict(np.array([10.0]))
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
```

This is a neural network :

tf.keras.Sequential([...])

- This creates a Sequential model, meaning layers are stacked one after another in a linear fashion.



tf.keras.layers.Dense(units=1) means:

- You're adding 1 neuron (just one circle) in that layer.

- Since it's a Dense layer, this one neuron is fully connected to all previous inputs.

model = tf.keras.Sequential([

tf.keras.Input(shape=(1,)), # One input feature → 1 green arrow

tf.keras.layers.Dense(units=1) # One output neuron → 1 blue circle

])

```
Copy
Input    →    Dense Layer    →    Output
  O          →     O         →     Y
```

More visually:

```
Copy
[ X ]  →  ( weight × X + bias )  →  [ Y ]
  O            →       O              →     O
```

If tf.keras.Input(shape=(2,)),

```
Copy
[ X1, X2 ]  →  ( W1 × X1 + W2 × X2 + b )  →  Y
    O   O            →              O            →   O
```

```
model.compile(optimizer='sgd', loss='mean_squared_error')
```

"Hey model, when you're learning, use this method to improve (sgd) and use this formula (mean_squared_error) to see how wrong you are."

optimizer='sgd' (Stochastic Gradient Descent)

- Tells the model how to learn.

- SGD = one of the simplest ways: guess → check → adjust.

- Think of it like trial and error, but smart.

loss='mean_squared_error'

- Tells the model how to measure mistakes.

- If the model guesses 4 but the answer is 6, the error is 2 — this squares it to punish big mistakes more.

```
model.fit(xs, ys, epochs=500)

model.predict(np.array([10.0]))
```

model.fit(xs, ys, epochs=500)

- Train the model using the data xs and ys

- Do it 500 times (called “epochs”) to learn the pattern well

- After training, the model will figure out the best math formula (like Y = 2X - 1)

model.predict(np.array([10.0]))

- Ask the model: “If X is 10, what’s Y?”

- The model uses what it learned to predict Y