CSE422: Artificial intelligence

How Machines Learn

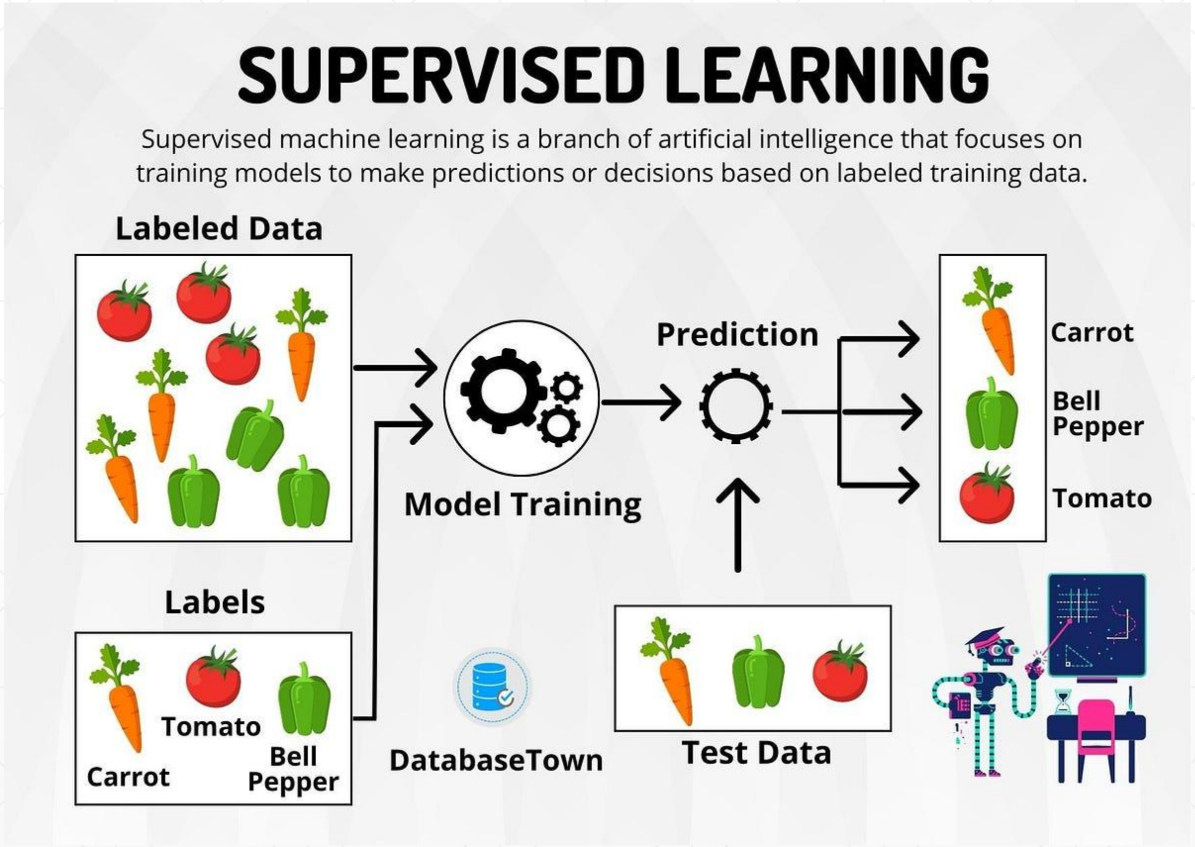
Linear Regression, Logistic Regression, Gradient Descent

Asif Shahriar
Lecturer, CSE, BRACU

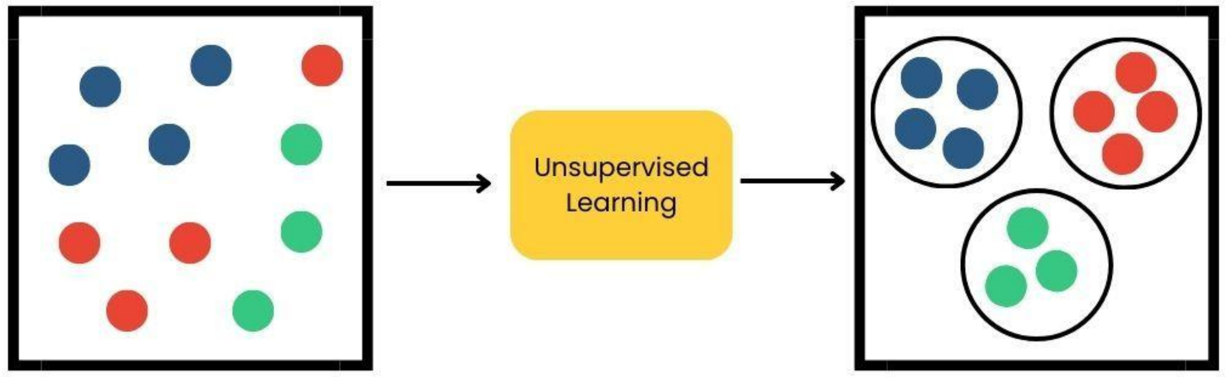
Machine Learning Tasks

- Machine learning tasks can be divided into three broad categories
- **Supervised Learning**
  - Learns from **labeled** training data (input-output pairs)
  - The model “supervises” itself by making predictions for input data and comparing the with known answers (right: okay, wrong: improve)
  - Used for prediction tasks like spam detection, price forecasting
- **Unsupervised Learning**
  - Works with **unlabeled** data to find patterns, groups or structures
  - Example: clustering (grouping customers into segments based on buying behavior), dimensionality reduction, anomaly detection
- **Reinforcement Learning**
  - The algorithm learns by interacting with an environment, making decisions, and receiving **feedback** in the form of **rewards** or **penalties**
  - Applied in robotics, game playing, and self-driving cars

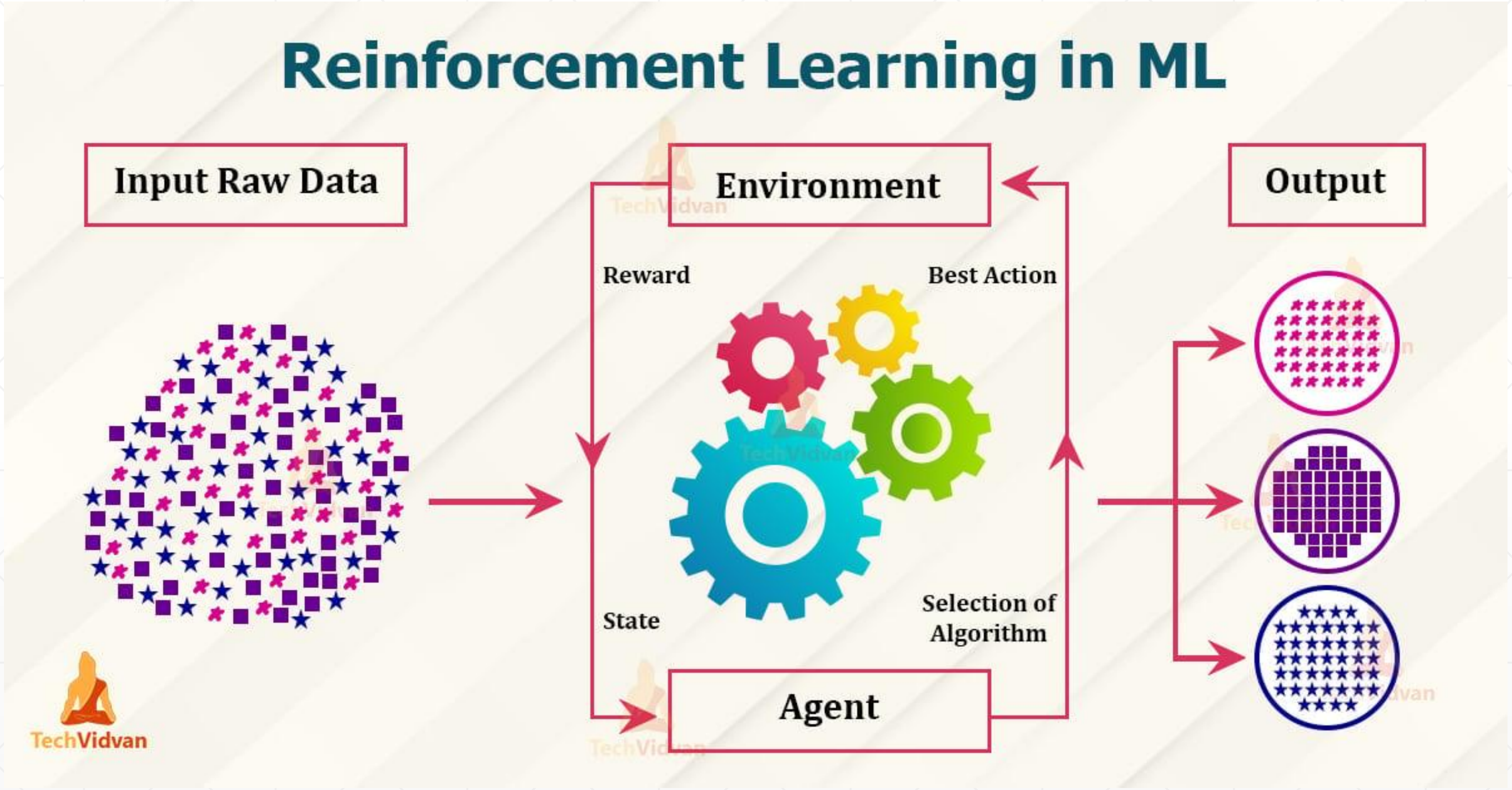
Supervised Learning


SUPERVISED LEARNING
Supervised machine learning is a branch of artificial intelligence that focuses on training models to make predictions or decisions based on labeled training data.
Labeled Data
Labels
Tomato
Carrot
Bell Pepper
Model Training
Prediction
Carrot
Bell Pepper
Tomato
Test Data
DatabaseTown

Unsupervised Learning



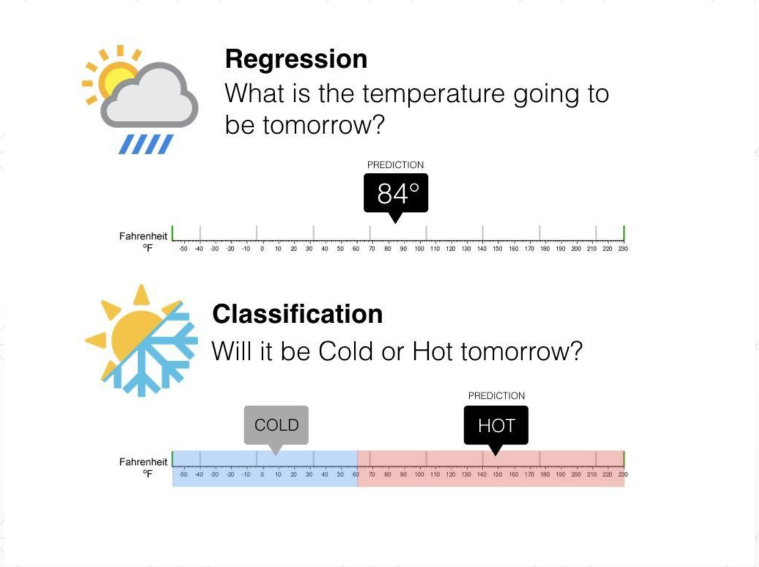
Reinforcement Learning

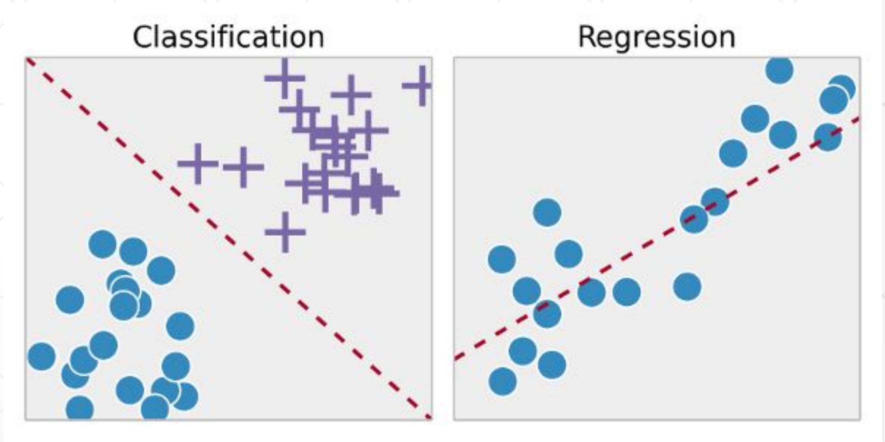


Supervised Learning

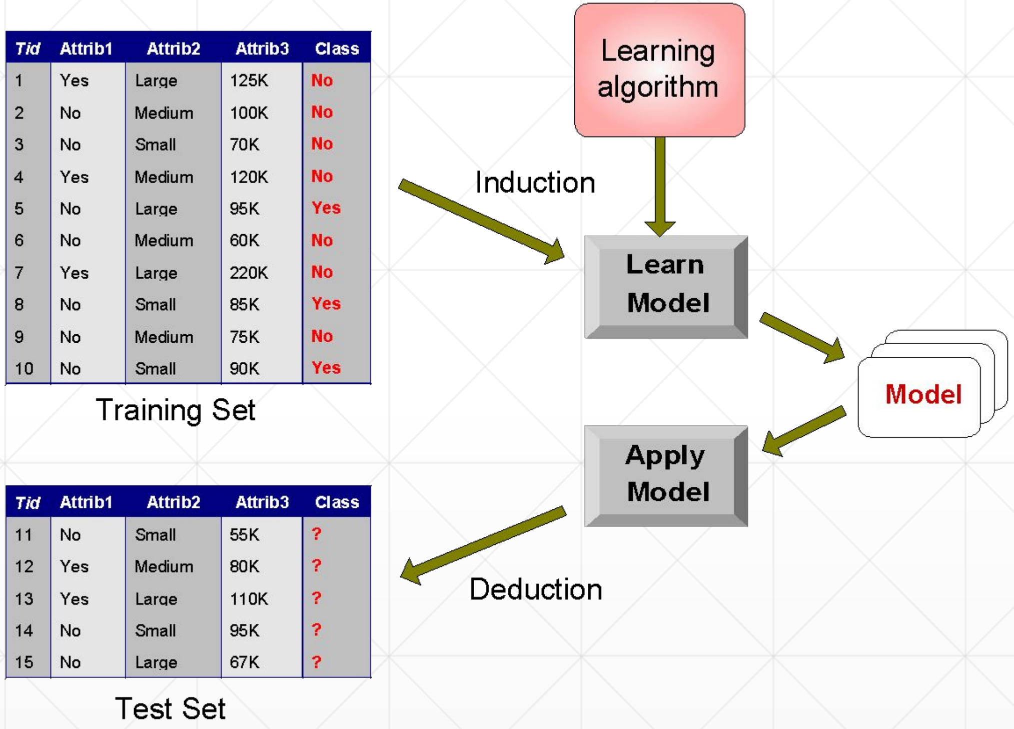
- Supervised Learning can be of TWO types
- **Regression**
  - Predicts a **continuous numerical value** based on input features
  - Example: Estimating house price from size, location, and number of rooms.
- **Classification**
  - Predicts a **discrete class label** from input data
  - Example: Classifying an email into <spam, ham>, identifying emotion from a social media post <happy, sad, angry, fearful, sarcastic>

Supervised Learning





Train-Test Set



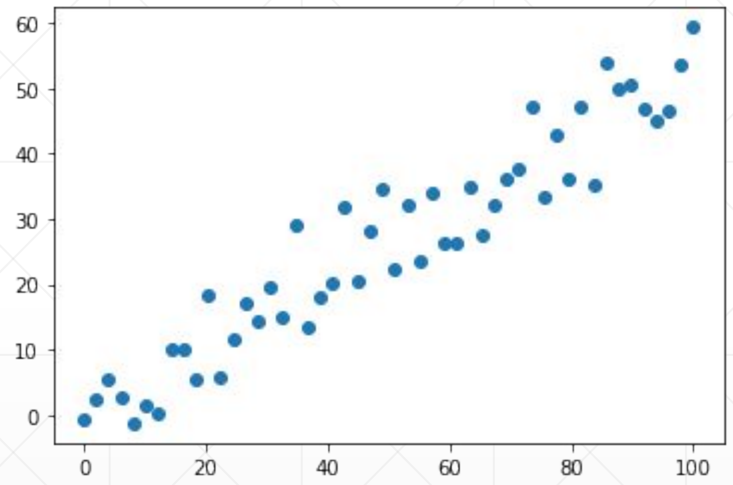
Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Linear Regression



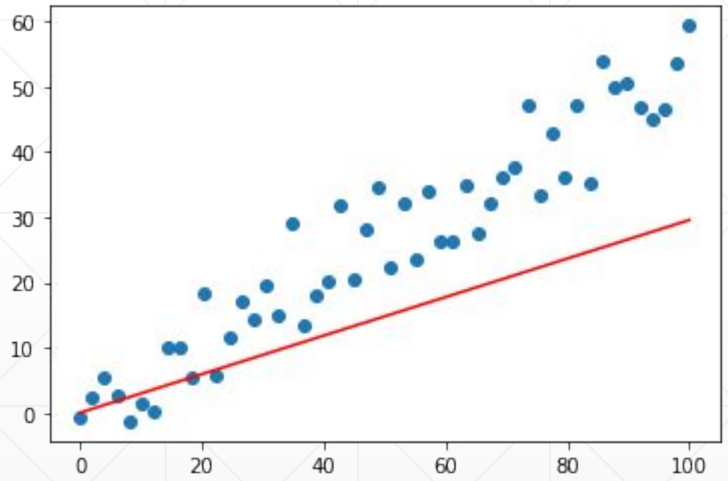
Linear Regression

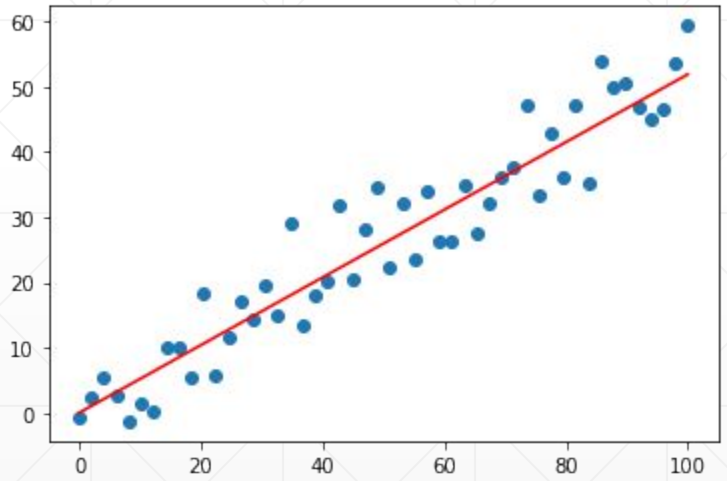
- The most straightforward (and very powerful) regression model is a Linear model
- Here we assume that the features (x) and the target (y) have a linear relationship
- In other words, we try to **FIT** a straight line on the dataset
- For each input x_i, we predict the output (target) as $\mathbf{h_w(x) = w_0 + w_1 x}$
- w_0 and w_1 are called **model parameters** or **weights**, we need to find their optimal values during training
- w_0 (the constant term) is often called **"bias"**
- We compare our prediction $h_w(x)$ with the actual output y to check how right or wrong our current model is

Linear Regression

Here are TWO linear regression models (straight lines) fit on the same data.

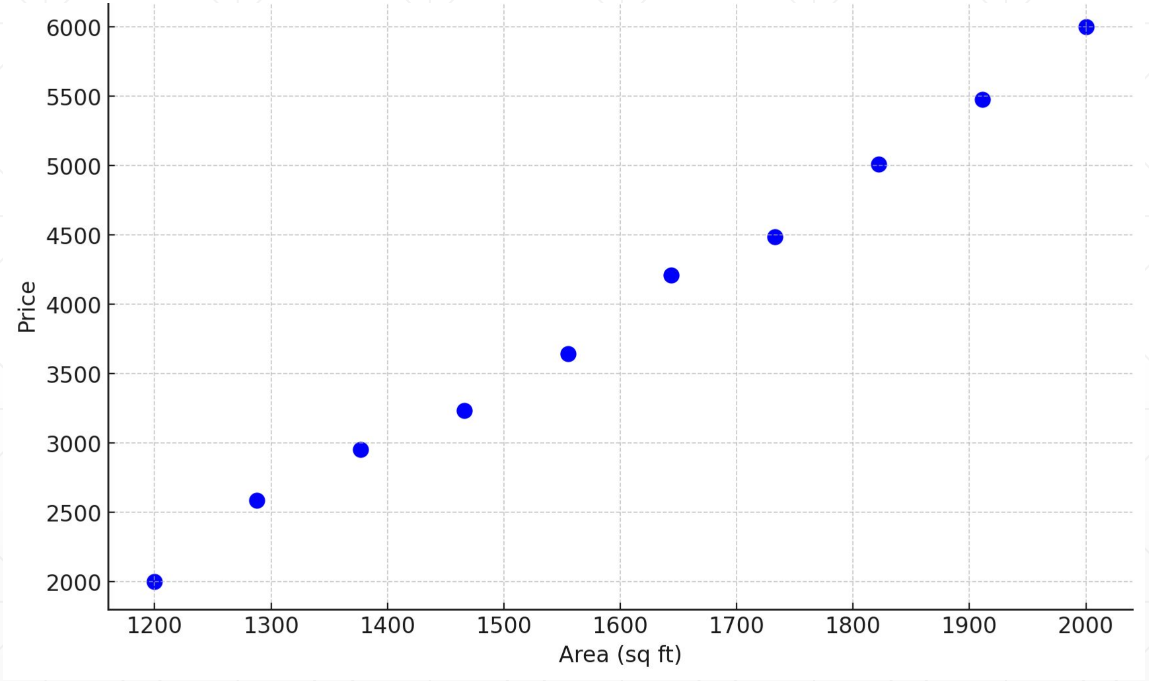
Which model is better?





A Mini-Housing Dataset

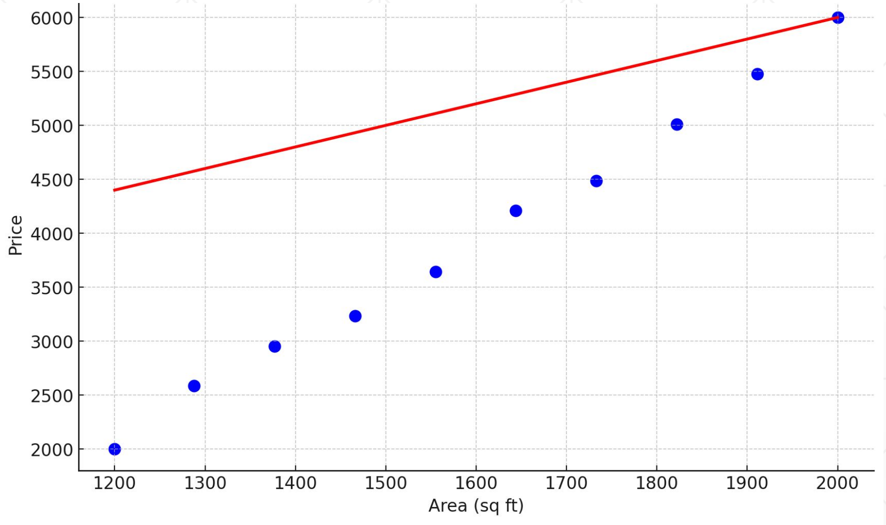
Area (x)	Price (y)
1200	2000
1288	2588
1377	2955
1466	3236
1555	3646
1644	4208
1733	4485
1822	5012
1911	5476
2000	6000



Linear Regression: Model Building

Let our initial model be ***h(x) = 2x + 2000***

- x: feature
- h(x): predicted output (price)
- y: actual output (price)
- Here $w_0 = 2000$, $w_1 = 2$



Area (x)	Price (y)	Predicted= $h_w(x)$
1200	2000	4400
1288	2588	4576
1377	2955	4754
1466	3236	4932
1555	3646	5110
1644	4208	5288
1733	4485	5466
1822	5012	5644
1911	5476	5822
2000	6000	6000

Linear Regression: Loss Function

- After we get predictions using our initial linear model, we need to determine the **“goodness”** of the current model (how good the model performs)
- The most common approach is to use a **loss function**
- A loss function measures how **bad** the current model is
- Our goal is to **minimize the loss function**
- The choice of loss function depends on the problem. For regression, we can use the following loss functions:
    - **Absolute Error (AE): $|y - h_w(x)|$**
        - In practice we use **Mean Absolute Error (MAE): $|y - h_w(x))| / N$**
    - **Squared Error (SE): $(y - h_w(x))^2$**
        - In practice we use **Mean Squared Error (MSE): $(y - h_w(x))^2 / N$**
        - Also called L2 loss function

Linear Regression: Loss Function

Area (x)	Price (y)	Pred=h(x)	AE	MAE	SE	MSE
1200	2000	4400	2400	2400	5760000	5760000
1288	2588	4576	1988	2194	3952144	4856072
1377	2955	4754	1799	2062.333	3236401	4316182
1466	3236	4932	1696	1970.75	2876416	3956240
1555	3646	5110	1464	1869.4	2143296	3593651
1644	4208	5288	1080	1737.833	1166400	3189110
1733	4485	5466	981	1629.714	962361	2871003
1822	5012	5644	632	1505	399424	2562055
1911	5476	5822	346	1376.222	119716	2290684
2000	6000	6000	0	1238.6	0	2061616
			12386	1238.6	20616158	2061616

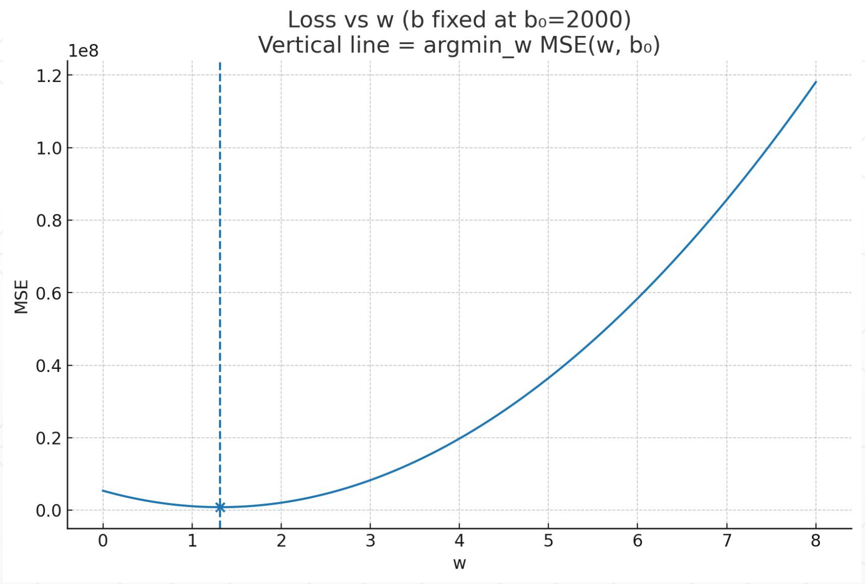
Linear Regression: Model Update

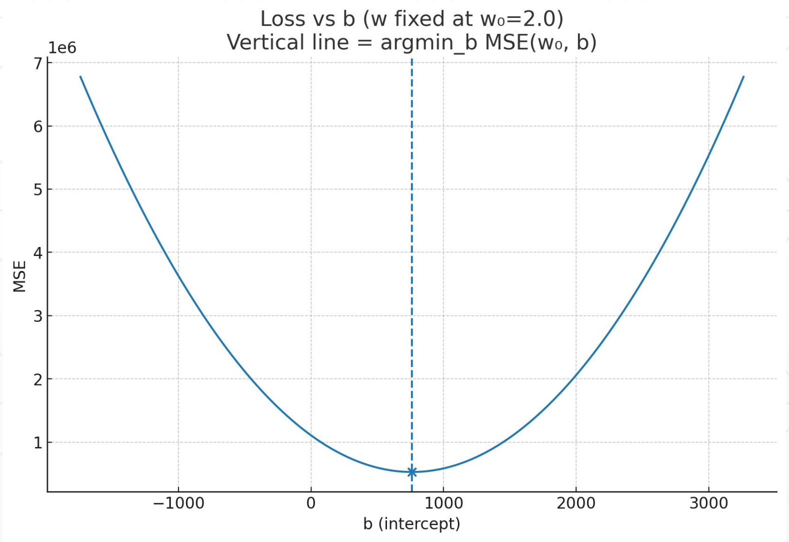
- Once we have the loss/error values, we need to update the model to make it better
- This is done by updating the model parameters
- We want to select new values for our parameters that minimizes the current loss function
    - For this we use Gradient Descent

Gradient Descent

We want to choose new values for w_0 and w_1 that minimize the L2 loss function (MSE):

$$(y - h_w(x))^2 / N$$





Gradient Descent

- $w* = \text{argmin}_w \text{Loss}(h_w)$
- Set partial derivatives to 0 to obtain the minima [recall HSC calculus]

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

$$\frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

Gradient Descent

$$\begin{aligned}\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x)) \\ &= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)).\end{aligned}$$

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2\left(y - h_{\mathbf{w}}(x)\right)$$

$$\frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2\left(y - h_{\mathbf{w}}(x)\right) \times x$$

Gradient Descent

```
$\mathbf{w} \leftarrow$ any point in the parameter space
while not converged do
    for each $w_i$ in $\mathbf{w}$ do
        $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$
```

For single example:

$$w_0 \leftarrow w_0 + \alpha\ (y - h_{\mathbf{w}}(x))$$

$$w_1 \leftarrow w_1 + \alpha\ (y - h_{\mathbf{w}}(x)\ ) \times\ x$$

For entire training set:

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))\ \times\ x_j$$

Performing Gradient Descent

Area (x)	Price (y)	Pred=h(x)	y-h(x)	[y-h(x)]x
1200	2000	4400	-2400	-2880000
1288	2588	4576	-1988	-2560544
1377	2955	4754	-1799	-2477223
1466	3236	4932	-1696	-2486336
1555	3646	5110	-1464	-2276520
1644	4208	5288	-1080	-1775520
1733	4485	5466	-981	-1700073
1822	5012	5644	-632	-1151504
1911	5476	5822	-346	-661206
2000	6000	6000	0	0
Sum			-12386	-17968926

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

Let $\alpha = 1e - 8$

w_0 = 2000 + (1e-8)(-12386) = 1999.999987

w_1 = 2 + (1e-8)(-17968926) = 1.82

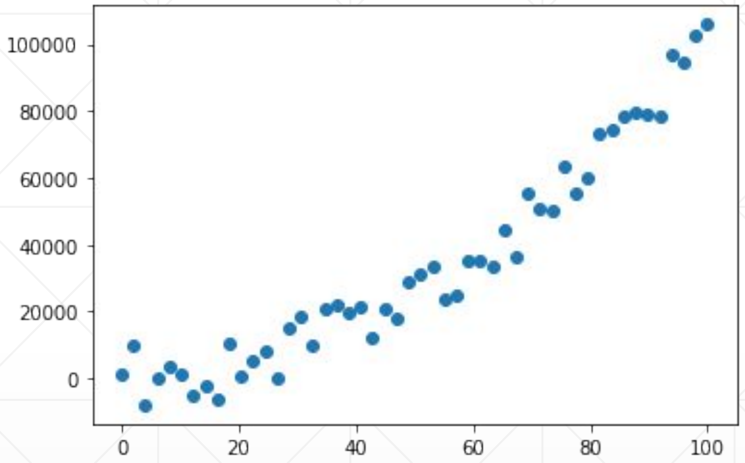
Multi-variable Linear Regression

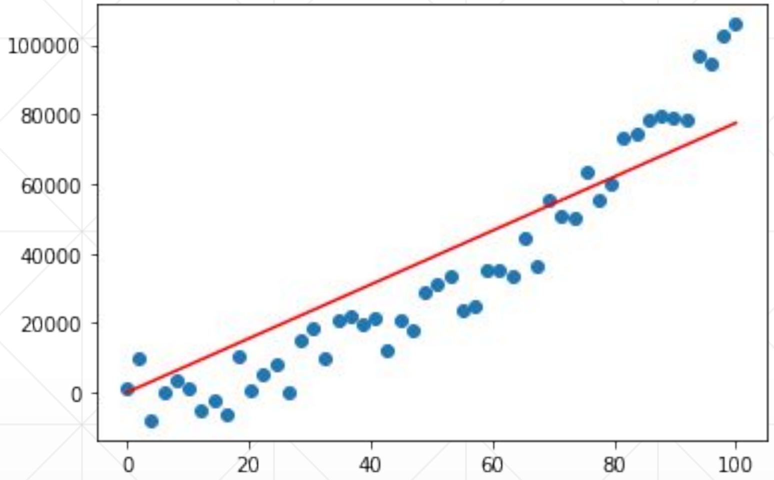
- Linear regression is not limited to a single feature only
- If there are multiple features, we can extend Linear Regression model naturally:

$$\boxed{h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i}}$$

- We can use Linear Regression to model non-linear relationships too!

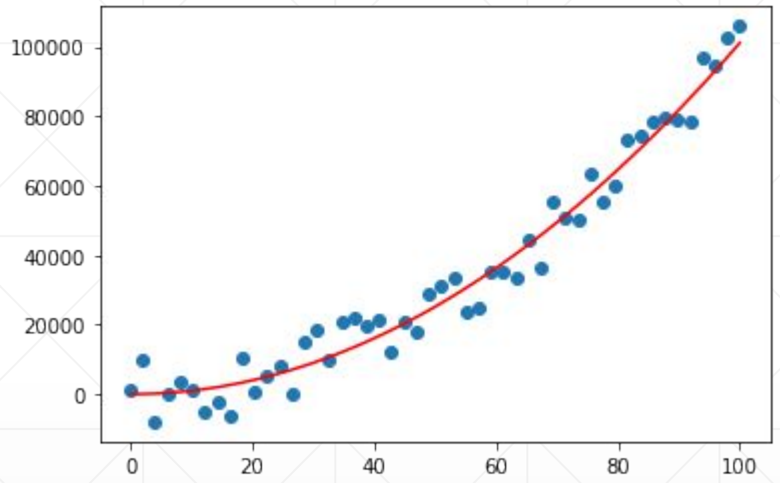
Linear Regression for Non-Linear Relations



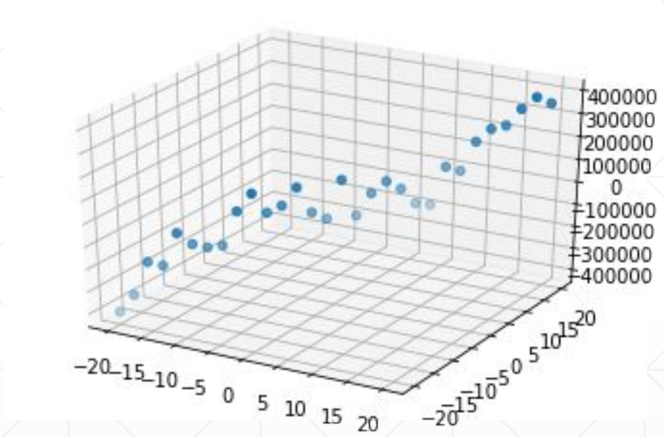


- A linear function (straight line) does not quite fit
- Just extend the model! $\mathbf{h_w(x) = w_0 + w_1 x + w_2 x^2}$

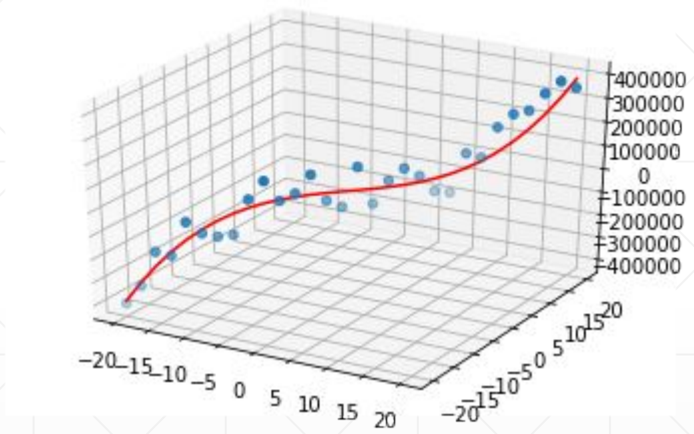
Linear Regression for Non-Linear Relations



Linear Regression for Non-Linear Relations



Linear Regression for Non-Linear Relations



But what about Classification?

Hours Study	Pass
2	No
3	No
4	Yes
5	No
6	Yes
7	Yes
8	Yes
9	Yes
10	?

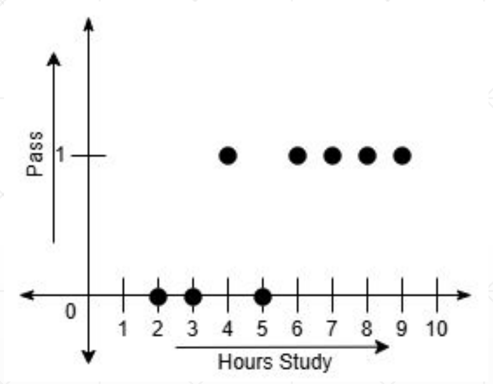
$\longrightarrow$

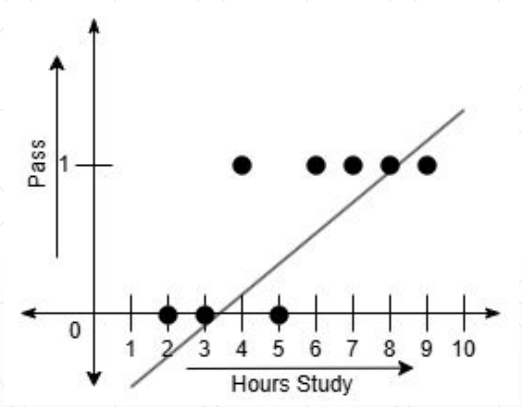
Hours Study	Pass
2	0
3	0
4	1
5	0
6	1
7	1
8	1
9	1
10	?

- Here, let's say we are trying to predict whether a student is going to pass based on hours study
- Hours study, in this case is the feature and Pass, is a label
- Pass is a categorical variable consisting of two values {Yes, No}
- This is a **classification** problem [binary classification]

But what about Classification?

- Let’s try to plot the dataset





Hours Study	Pass
2	0
3	0
4	1
5	0
6	1
7	1
8	1
9	1
10	?

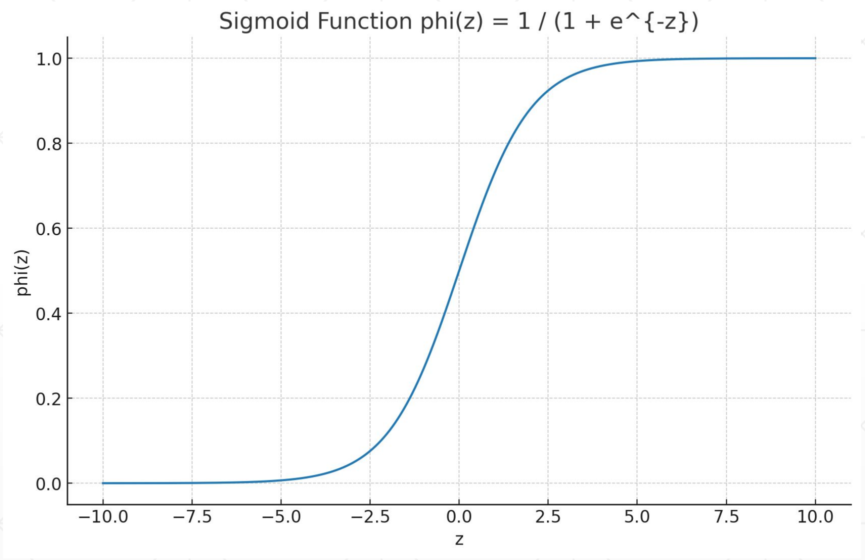
- Does not fit
- Linear models cannot fit non-linear classification data

Linear Model for Classification?

- So how to fit a linear model to non-linear classification problem?
    - By passing the linear output through a **non-linear activation function**
- For binary classification, a suitable non-linear activation function is **sigmoid function**
    - **Linear function: $z = w_0 + w_1 x$**
    - **Sigmoid function: $\sigma(z) = 1 / (1 + e^{-z})$ [$\sigma(z)$** is called **Logit]**
    - This is called **Logistic Regression** (note: this is NOT actually regression, it is classification. We keep the regression word because the idea originally came from Linear Regression)

Sigmoid Function

z	σ(z)	z	σ(z)
-10	4.50E-05	1	0.731059
-9	0.000123	2	0.880797
-8	0.000335	3	0.952574
-7	0.000911	4	0.982014
-6	0.002473	5	0.993307
-5	0.006693	6	0.997527
-4	0.017986	7	0.999089
-3	0.047426	8	0.999665
-2	0.119203	9	0.999877
-1	0.268941	10	0.999955
0	0.5		



Logistic Regression for Classification

- **$z = w_0 + w_1 x$**
- Using sigmoid function, we get the **logits**: $\phi(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$
- For binary classification, these logits are compared with a fixed threshold, i.e. 0.5
  - If Φ(z) > threshold, prediction y' = 1, else y' = 0
  - Compare the prediction y' against the true label, y
- For binary classification, loss is calculated using **Binary Cross-Entropy (BCE) loss**

$$l = y \ \log[\phi(z)] + (1 - y) \ \log[1 - \phi(z)]$$

Intuition behind BCE Loss Function

- If y = 1 and σ(z) = 0.000005 [Extreme case of misclassification]
  - loss = -1.log0 - (1-1).log(1-0) = 5.301 [High loss value]
- If y = 1 and a = 1 [Best case of correct classification]
  - loss = -1.log1 - (1-1).log(1-1) = 0 [Low loss value]

- If y = 0 and σ(z) = .000095 [Extreme case of misclassification]
  - loss = -0.log1 - (1-0).log(1-1) = 4.022 [High loss value]
- If y = 0 and a = 0 [Best case of correct classification]
  - loss = -0.log0 - (1-0).log(1-0) = 0 [Low loss value]

Logistic Regression for Classification

- **Updating parameters using gradient descent:**

$$l = y\ \log[\phi(z)] + (1-y)\ \log[1-\phi(z)]$$

$$\frac{\delta l}{\delta z} = y \times \frac{1}{\phi(z)}\phi'(z) + (1-y) \times \frac{1}{1-\phi(z)}\left(-\phi'(z)\right)$$

$$= \left[y \times \frac{1}{\phi(z)} - (1-y) \times \frac{1}{1-\phi(z)}\right]\phi'(z)$$

$$= \left[y \times \frac{1}{\phi(z)} - (1-y) \times \frac{1}{1-\phi(z)}\right]\phi(z)[1-\phi(z)]$$

$$= y \times [1-\phi(z)] - (1-y)\phi(z)$$

$$= y - \phi(z)$$

Logistic Regression for Classification

- Since **z = w$_0$ + w$_1$ x,**

$$\frac{\delta z}{\delta w_0} = 1, \quad \frac{\delta z}{\delta w_1} = x$$

- Using chain rule,

$$\frac{\delta l}{\delta w_0} = \frac{\delta l}{\delta z} \times \frac{\delta z}{\delta w_0} = y - \phi(z)$$

$$\frac{\delta l}{\delta w_1} = \frac{\delta l}{\delta z} \times \frac{\delta z}{\delta w_1} = [y - \phi(z)]\, x$$

- Does it look familiar?
  - Similar to linear regression

Logistic Regression for Classification

- For single data point:

$$w_0 \leftarrow w_0 + \alpha\,[y - \phi(z)]$$

$$w_1 \leftarrow w_1 + \alpha\,[y - \phi(z)]\,x$$

- For entire training set:

$$w_0 \leftarrow w_0 + \alpha \sum_j [y - \phi(z)]$$

$$w_1 \leftarrow w_1 + \alpha \sum_j [y - \phi(z)]\,x_j$$

Summary of Logistic Regression Steps

- Take $\mathbf{z = w_0 + w_1 x}$ with randomly initialized w_0 and w_1
- Pass through sigmoid function to get the logit **σ(z)** [for Binary classification]
- Compare the logit against threshold to get the predicted value y'
- Compare y' with y, calculate the loss
- Apply gradient descent to update w_0 and w_1
- Repeat

Logistic Regression for Classification

- You are training a logistic regression model with two input features
- The prediction is: $y' = \sigma(z)$ where $z = w_1 x_1 + w_2 x_2 + b, \sigma(z) = 1/\ 1 + e^{-z}$
- Input features: $x_1 = 1$ and $x_2 = 2$
- Bias: b = 0
- True label: y = 1
- Initial weights: $w_1 = 0.2$ and $w_2 = -0.4$
- Learning rate, $\alpha = 0.1$

Based on these, answer the following questions:

1. Compute the value of z and the predicted output y'.
2. Using Binary Cross-entropy loss given as $\frac{\partial Loss}{\partial w_i} = (y' - y)x_i$ perform one step of gradient descent to update the weights.