machine learning as the field of study that gives computers the ability to learn without being explicitly programmed.

3-4 types of machine learning

- Supervised machine learning
- Unsupervised machine learning
- Recommender systems
- reinforcement learning

you will first train your model with examples of inputs x and the right answers, that is the labels y. After the model has learned from these input, output, or x and y pairs, they can then take a brand new input x, something it has never seen before, and try to produce the appropriate corresponding output y.

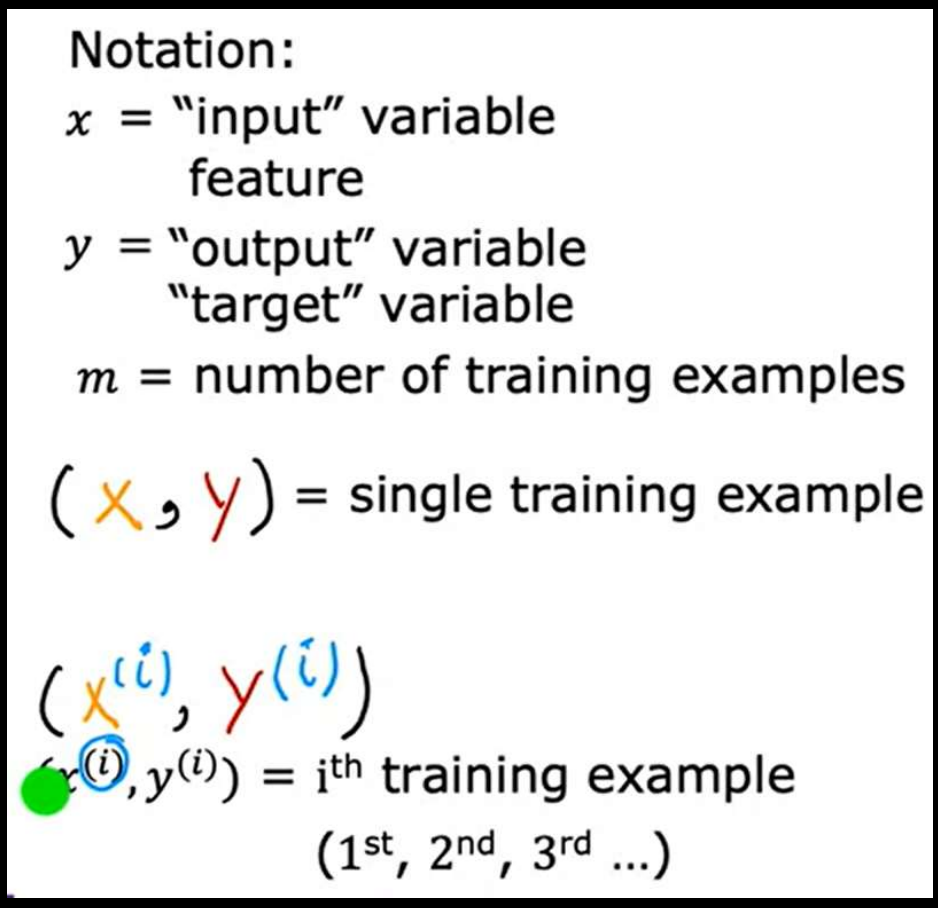
The two major types of supervised learning are regression and classification. In a regression application like predicting prices of houses, the learning algorithm has to predict numbers from infinitely many possible output numbers. Whereas in classification the learning algorithm has to make a prediction of a category, all of a small set of possible outputs.

Unsupervised learning(part 1): unsupervised learning, we call it unsupervised because we're not trying to supervise the algorithm. To give some quote the right answer for every input, instead, we asked our room to figure out all by ourselves what's interesting. Or what patterns or structures that might be in this data, with this particular data set.
clustering algorithm. Because it places the unlabeled data into different clusters and this turns out to be used in many applications.
to summarize a clustering algorithm. Which is a type of unsupervised learning algorithm, *takes data without labels and tries to automatically group them into clusters.*

Part 2:

1. a clustering algorithm, which groups similar data points together.
2. anomaly detection, which is used to detect unusual events.
3. dimensionality reduction. This lets you take a big data-set and almost magically compress it to a much smaller data-set while losing as little information as possible.

The dataset that is used to train the model is called a **training set.**



Cost function formula:

$$f_{w,b}(x) = wx + b$$

$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$

$f_{w,b}(x^{(i)}) = wx^{(i)} + b$

Find w, b:
$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

Cost function: Squared error cost function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

$\hat{y}^{(i)}$: error

m = number of training examples

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right)^2$$

$J(w,b)$: intuition

ML:Gradient Descent

Gradient descent algorithm

Repeat until convergence

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

Learning rate

Derivative

Simultaneously update w and b

Correct: Simultaneous update

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$w = tmp\_w$$

$$b = tmp\_b$$

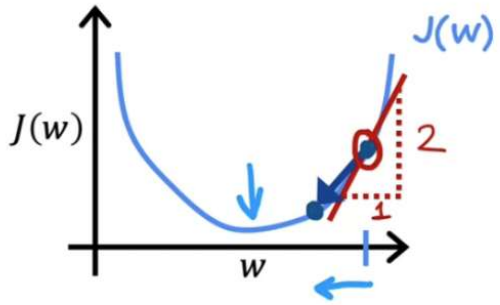
Incorrect

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$w = tmp\_w$$

$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

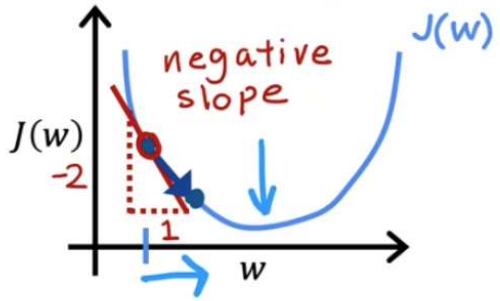
$$b = tmp\_b$$



$$w = w - \alpha \frac{d}{dw} J(w)$$

> 0

$$w = w - \alpha \cdot (positive\ number)$$



$$\frac{d}{dw} J(w) < 0$$

$$w = w - \alpha \cdot (negative\ number)$$

$$w = w - \alpha \frac{d}{dw} J(w)$$

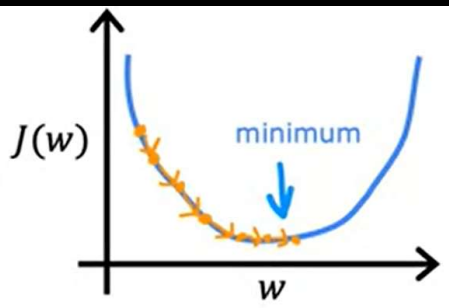
If α is too small...

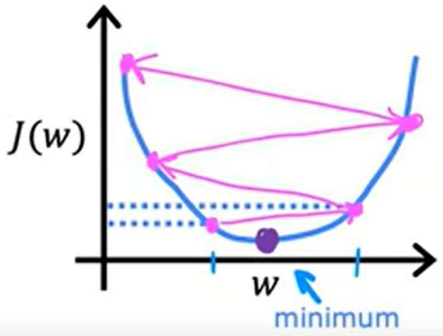
Gradient descent may be slow.

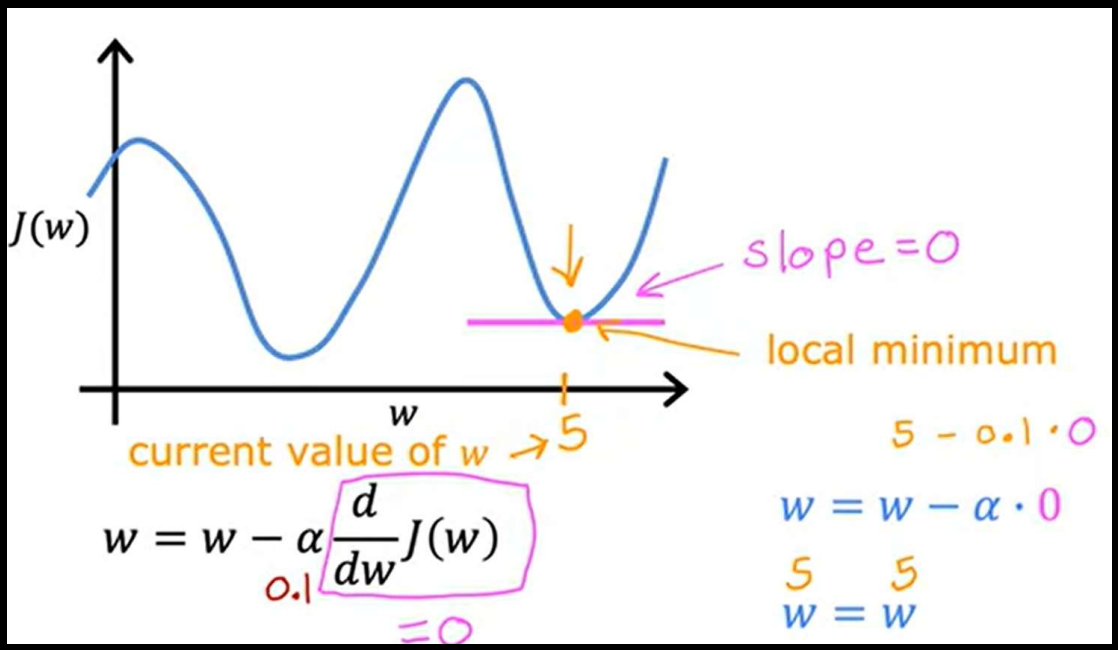
If α is too large...

Gradient descent may:

- Overshoot, never reach minimum
- Fail to converge, diverge







Near a local minimum,
- Derivative becomes smaller
- Update steps become smaller

Can reach minimum without decreasing learning rate α

(Optional)

$$\frac{\partial}{\partial w} J(w,b) = \frac{d}{dw} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 = \frac{d}{dw} \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)}\right) 2x^{(i)} = \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

$$\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)}\right) 2 = \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)$$

no $x^{(i)}$

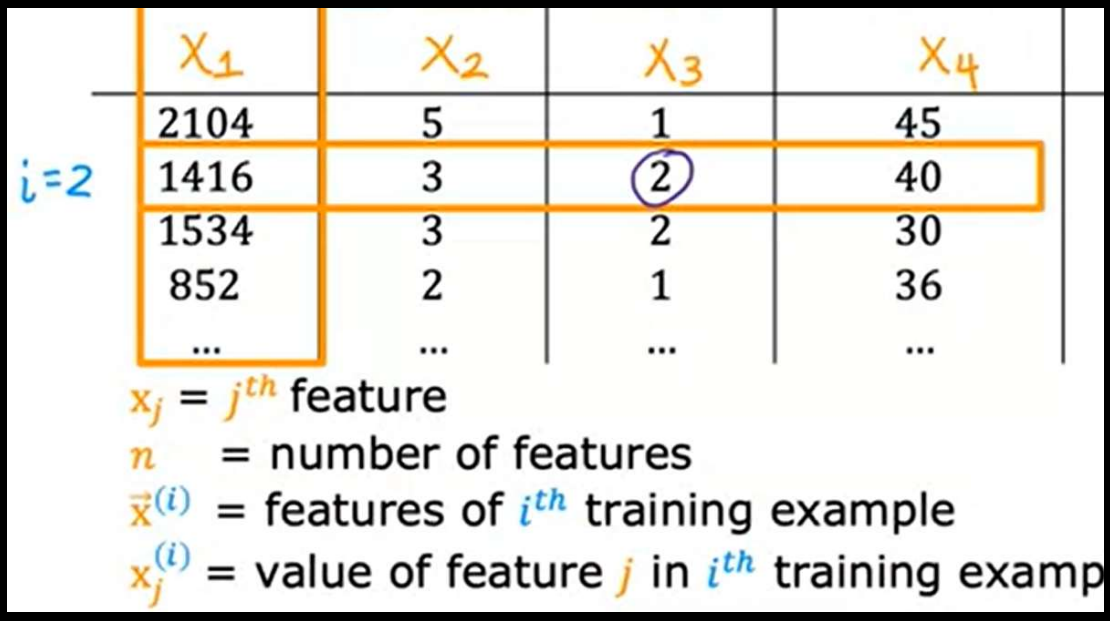
Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) \quad x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

}



$x_j^{(i)}$ = value of feature j in the i^{th} training example

$x^{(i)}$ = the column vector of all the feature inputs of the i^{th} training example

m = the number of training examples

$n = \left|x^{(i)}\right|$; (the number of features)

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

$\vec{w} = [w_1 \; w_2 \; w_3 \ldots w_n]$ parameters of the model

b is a number

vector $\vec{x} = [x_1 \; x_2 \; x_3 \ldots x_n]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$$

dot product

multiple linear regression

Parameters and features

$\vec{w} = [w_1 \quad w_2 \quad w_3]$ $n=3$

b is a number

$\vec{x} = [x_1 \quad x_2 \quad x_3]$

linear algebra: count from 1

NumPy

w[0] w[1] w[2]

```
w = np.array([1.0,2.5,-3.3])
b = 4
x = np.array([10,20,30])
```

code: count from 0

Without vectorization

$n = 100,000$

$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

Without vectorization

$f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b$ $\sum_{j=1}^{n} \rightarrow j=1 \ldots n$ 1,2,3

range(0,n) → j = 0...n-1

```
f = 0      range(n)
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
```

Vectorization

$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

```
f = np.dot(w,x) + b
```

Gradient descent

$\vec{w} = (w_1 \quad w_2 \quad \cdots \quad w_{16})$ ~~b~~ parameters

derivatives $\vec{d} = (d_1 \quad d_2 \quad \cdots \quad d_{16})$

```
w = np.array([0.5, 1.3, … 3.4])
d = np.array([0.3, 0.2, … 0.4])
```

compute $w_j = w_j - \overbrace{0.1}^{\text{learning rate } \alpha} d_j$ for $j = 1 \ldots 16$

Without vectorization

$w_1 = w_1 - 0.1d_1$

$w_2 = w_2 - 0.1d_2$

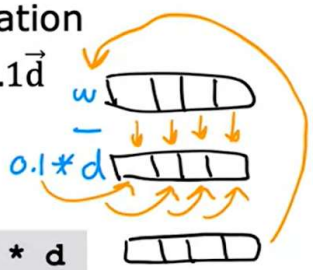
$\vdots$

$w_{16} = w_{16} - 0.1d_{16}$

```
for j in range(0,16):
    w[j] = w[j] - 0.1 * d[j]
```

With vectorization

$\vec{w} = \vec{w} - 0.1\vec{d}$




```
w = w - 0.1 * d
```

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	$\vec{w} = [w_1 \quad \cdots \quad w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ ← dot product
Cost function	$J(w_1, \cdots, w_n, b)$	$J(\vec{w}, b)$

Gradient descent

repeat {	repeat {
$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \cdots, w_n, b)$	$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$
$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \cdots, w_n, b)$	$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$
}	}

Gradient descent

One feature

repeat {

$$w = w - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}$$

$$\frac{\partial}{\partial w}J(w,b)$$

$$b = b - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)$$

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$j = 1$

$$w_1 = w_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_1^{(i)}$$

$$\vdots$$

$$\frac{\partial}{\partial w_1}J(\vec{w},b)$$

$j = n$

$$w_n = w_n - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_n^{(i)}$$

$$b = b - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$$

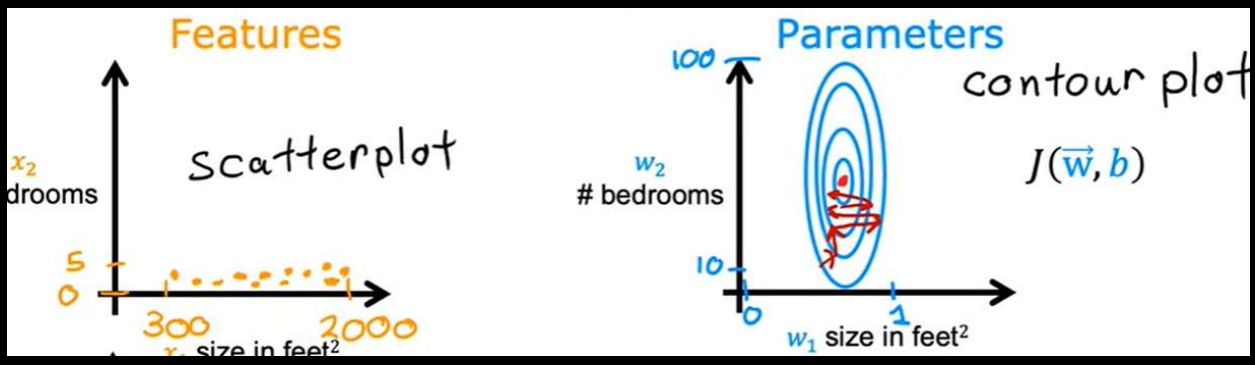
simultaneously update

w_j (for $j = 1, \cdots, n$) and b

}

Features and parameter values:

	size of feature x_j	size of parameter w_j
size in feet^2	⟷	⟷
#bedrooms	⟷	⟷



Feature scaling methods

$$300 \leq x_1 \leq 2000$$

$$x_{1,scaled} = \frac{x_1}{2000}$$

max

$$x_1 = \frac{x_1 - \mu_1}{2000 - 300}$$

max - min

$$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$$

σ_1

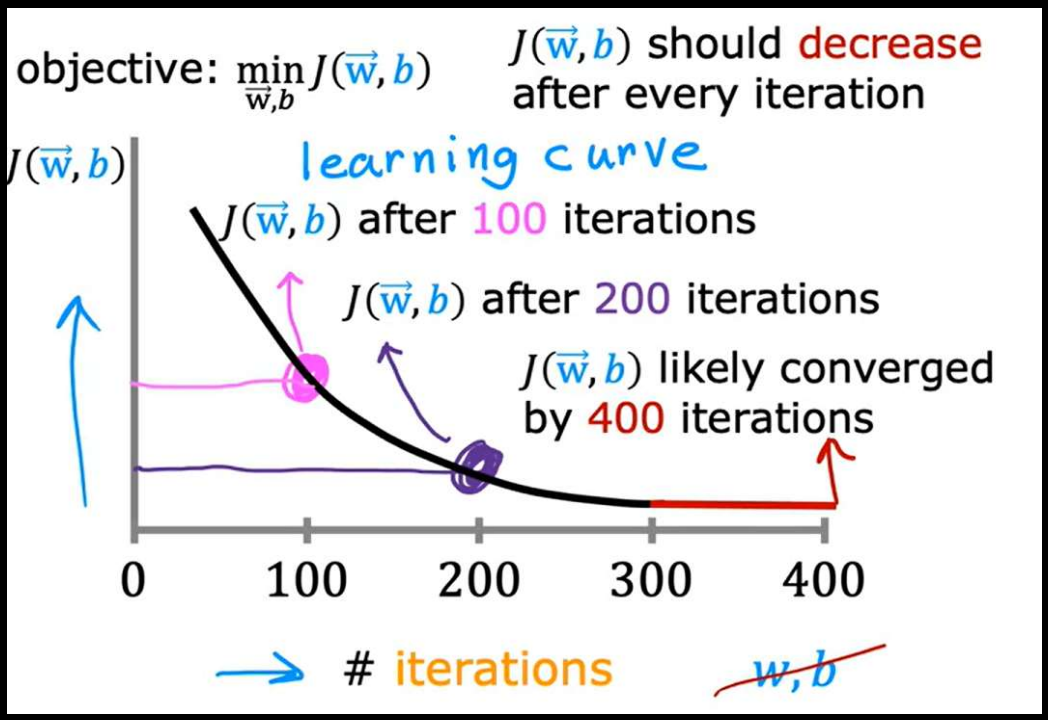
aim for about $-1 \le x_j \le 1$ for each feature x_j

$-3 \le x_j \le 3$

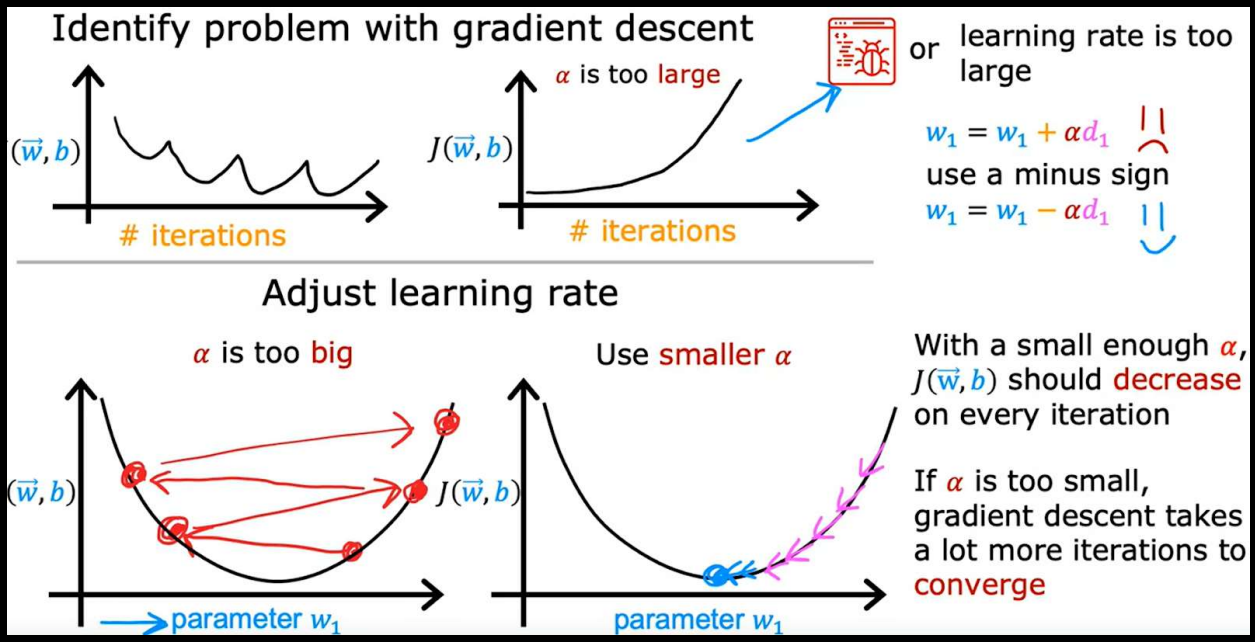
$-0.3 \le x_j \le 0.3$

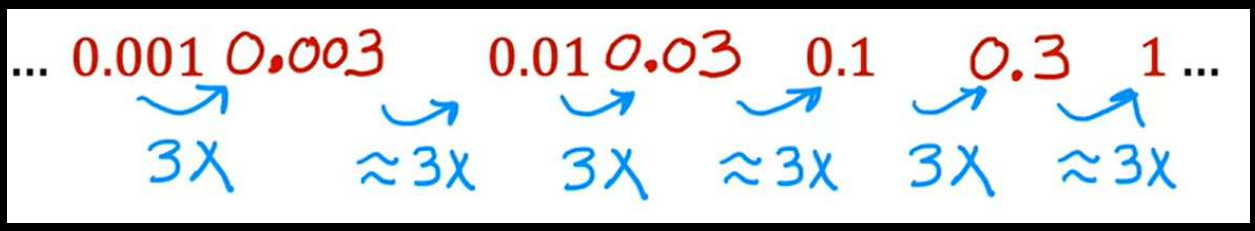
} acceptable ranges

$0 \le x_1 \le 3$	okay, no rescaling
$-2 \le x_2 \le 0.5$	okay, no rescaling
$-100 \le x_3 \le 100$	too large → rescale
$-0.001 \le x_4 \le 0.001$	too small → rescale
$98.6 \le x_5 \le 105$	too large → rescale


objective: $\min_{\vec{w},b} J(\vec{w},b)$
$J(\vec{w},b)$ should decrease after every iteration
$J(\vec{w},b)$
learning curve
$J(\vec{w},b)$ after 100 iterations
$J(\vec{w},b)$ after 200 iterations
$J(\vec{w},b)$ likely converged by 400 iterations
0
100
200
300
400
iterations
w, b

Learning rate(alpha)





Feature engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$

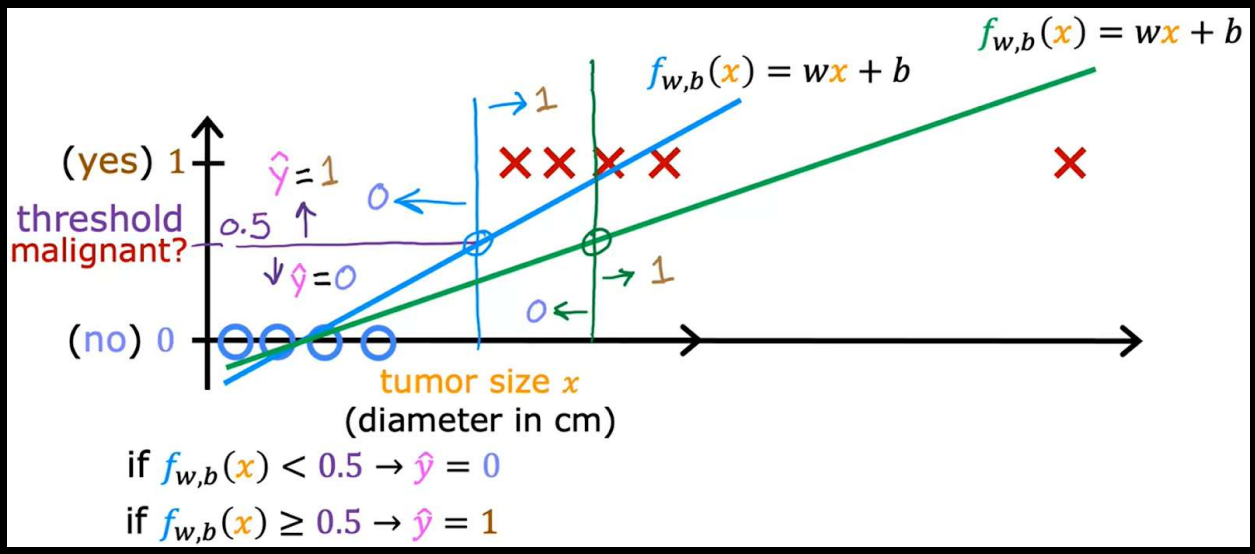
frontage depth

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$

new feature

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

$$g(z) = \frac{1}{1+e^{-z}}$$

$f_{\vec{w},b}(\vec{x})$

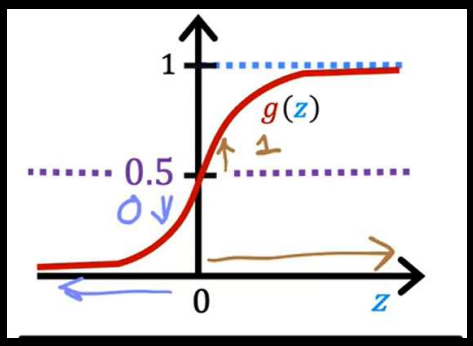
$$z = \vec{w} \cdot \vec{x} + b$$

$$z$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

"logistic regression"


1
0.5
0
g(z)
z

$f_{\vec{w},b}(\vec{x}) = g(\vec{w}\cdot\vec{x}+b) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

$= P(Y=1|\vec{x};\vec{w},b)$ to decide 0 or 1

If $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

Yes: $\hat{Y}=1$. No: $\hat{Y}=0$. | $\hat{y}$ = prediction

when $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(z) \geq 0.5$

$z \geq 0$

$\vec{w}\cdot\vec{x}+b \geq 0$ $\quad$ $\vec{w}\cdot\vec{x}+b < 0$

$\hat{Y}=1$ $\quad$ $\hat{y}=0$

decision boundary

for ex. taking & plotting a graphs that includes 2 features x_1 & x_2:

i.e; $f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1x_1+w_2x_2+b)$

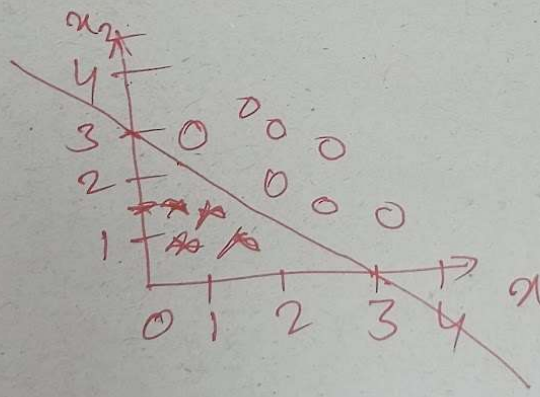
for ex. take $w_1=w_2=1$ & $b=-3$

decision boundary $z = \vec{w}\cdot\vec{x}+b=0$ | equal to zero

$z = x_1+x_2+b=0$

$x_1+x_2-3=0$

$x_1+x_2=3$



all the data points that come left are into one class & other into another class

there are non linear decision boundaries are also possible, where z is complex. for ex

$z = w_1x_1^2 + w_2x_2^2 + b$. this is a circle

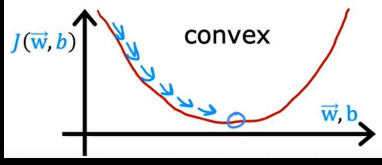
Squared error cost

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2$$

loss $L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)$

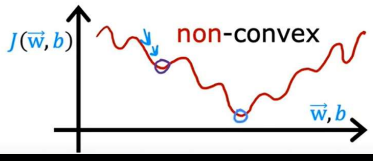
linear regression

$$f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$$



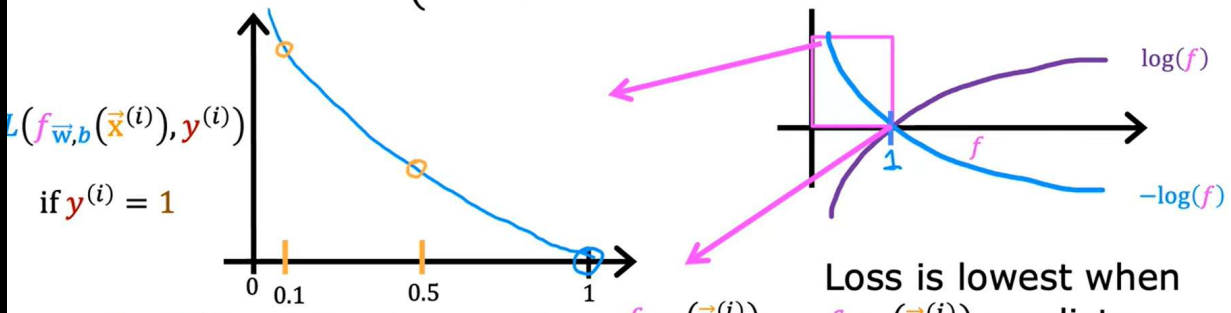
logistic regression

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$



Logistic loss function

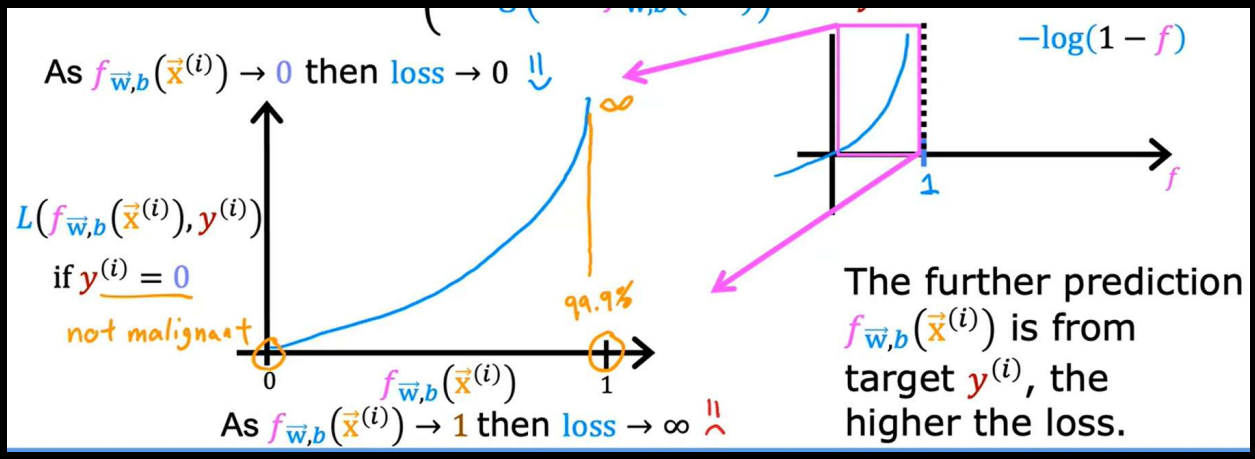
$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$



As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss $\to 0$

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to \infty$

Loss is lowest when $f_{\vec{w},b}(\vec{x}^{(i)})$ predicts close to true label $y^{(i)}$.



cost

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)}_{\text{loss}}$$

$$= \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$

convex
↳ can reach a global minimum

find w, b that minimize cost J

Simplified loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

$y^{(i)}$ → 0 ; $(1 - y^{(i)})$ → $(1-0)$

if $y^{(i)} = 1$:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -1 \log\left(f(\vec{x})\right)$$

if $y^{(i)} = 0$:

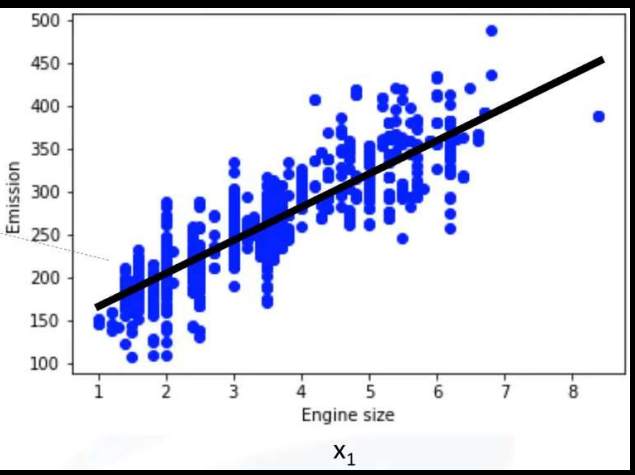
$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -(1-0)\log\left(1 - f(\vec{x})\right)$$

$$\hat{y} = \theta_0 + \theta_1 x_1$$

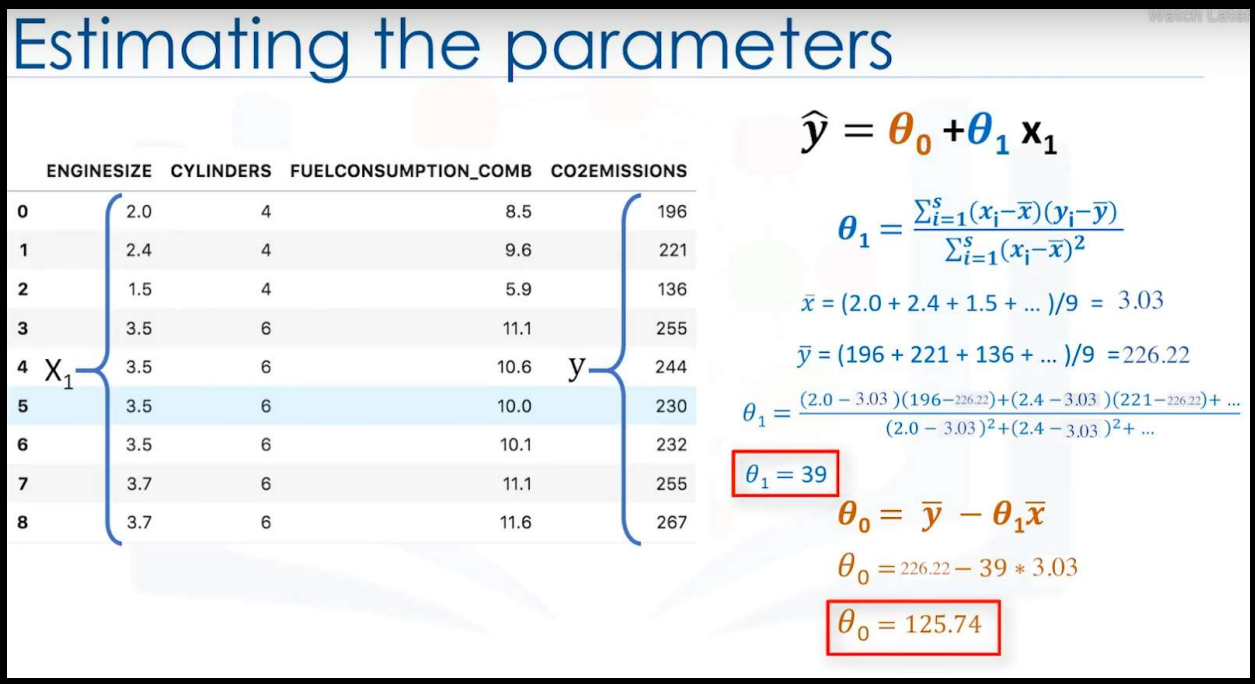
response variable

a single predictor

y



$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Model evaluation:

1. **The goal of regression is to build a model to accurately predict an unknown case.**

train and test on same data set

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	212

Test

y

Actual values

$$Error = \frac{(232 - 234) + (255 - 256) + \ ...}{4}$$

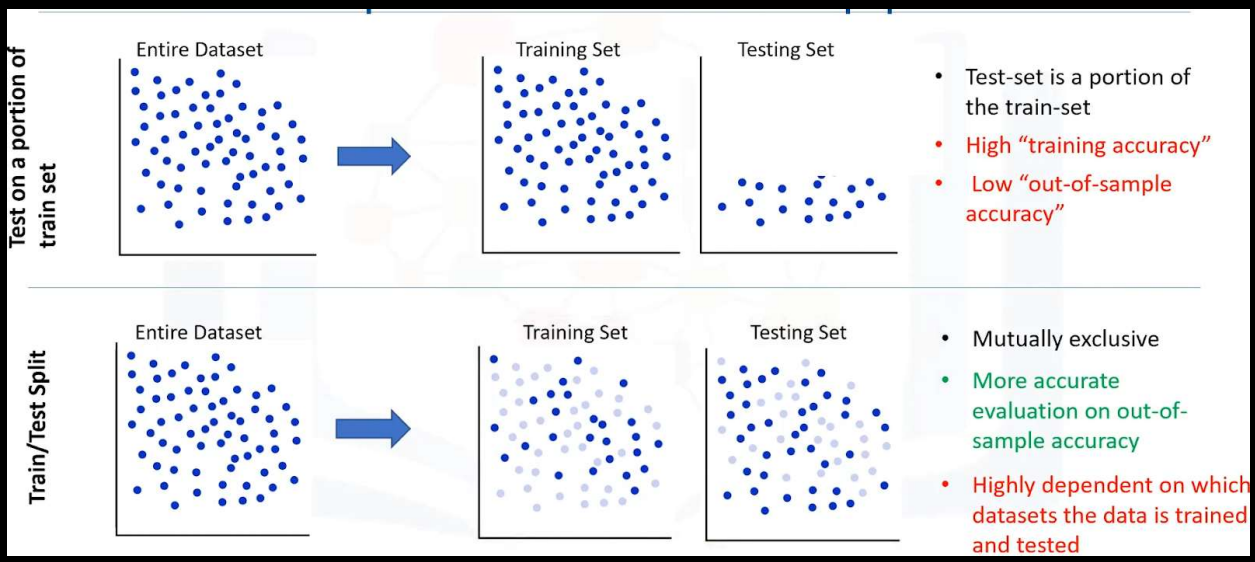
$$Error = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$\hat{y}$

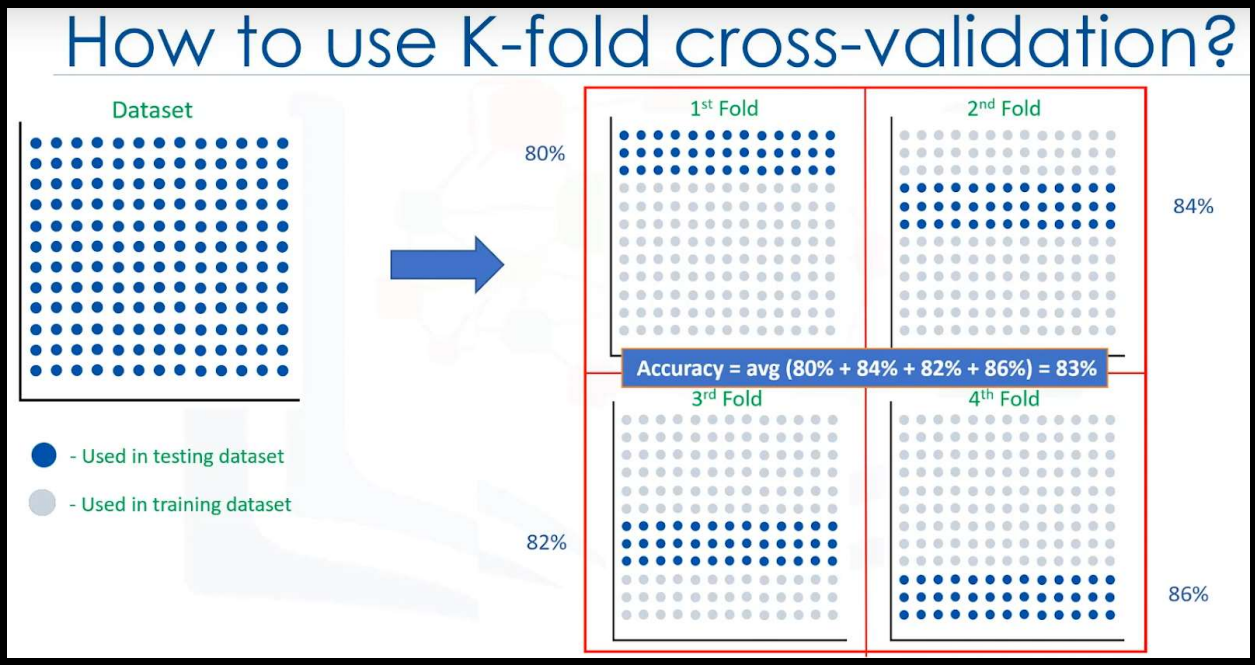
	Prediction
6	234
7	256
8	267
9	210

Predicted values

Train/test split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive.

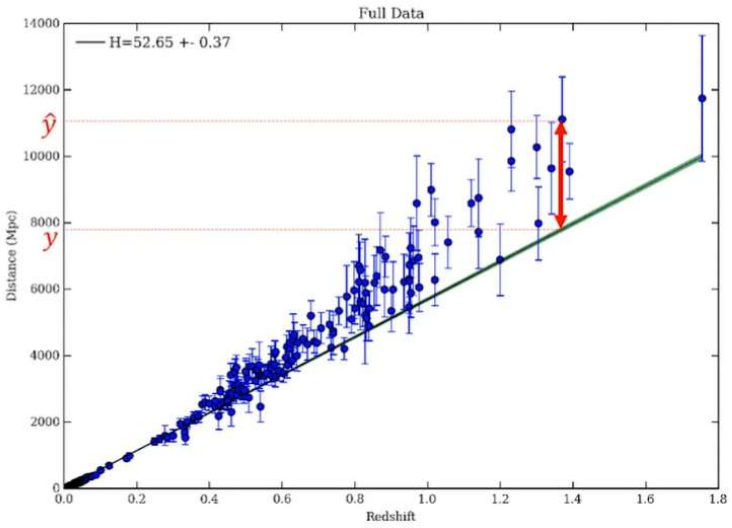


K-fold cross-validation



Accuracy metrics for model evaluation

What is an error of the model?



$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\mathrm{R}MSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$RAE = \frac{\sum_{j=1}^{n}|y_j - \hat{y}_j|}{\sum_{j=1}^{n}|y_j - \bar{y}|}$$

$$RSE = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y})^2}$$

$$R^2 = 1 - RSE$$