Evaluation Metrics

* Confusion Matrix →

Model witness diagnosis

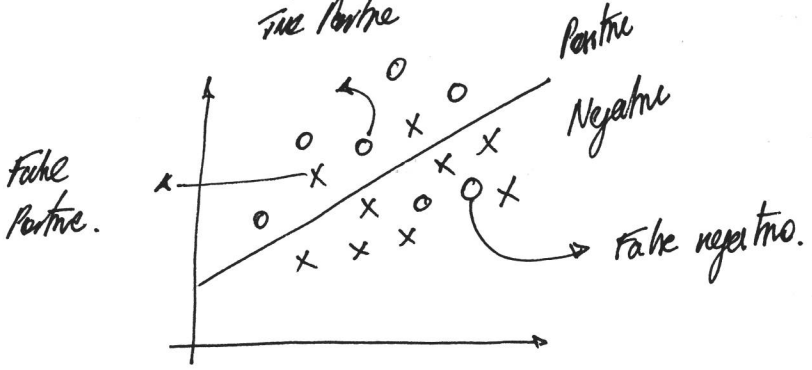
Possible outcomes →

	Diagnosed Sick	Diagnosed Healthy
Sick	True Positive	False Negative
Healthy	False Positive	True Negative

⟹

	Guessed Positive	Guessed Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

True Positive



* Accuracy →

Accuracy = ratio of number of good classifications / all points.

- When it doesn't work →

Credit card fraud → Good Transactions 284,335

Fraud 472

* If we say all are good → Acc = $\frac{284,335}{284,807} = 99.83\%$

Not a good model, it doesn't catch anything.

* False Negatives & Positives ⟶

Which one is worse depends on the context.

* Precision ⟶

	Diagnosed Sick	Diagnosed Healthy	← Critical field
Sick	1000	200	
Healthy	800	9000	

- Out of the patients that ~~we actually sick~~ we diagnosed sick how many are actually sick?

$$\text{Precision} = \frac{1000}{1.000 + 800} = 55.6\%$$

	Land is Spam	Land is Zahlen	← Critical field.
Spam	100	170	
Not spam	30	700	

$$\text{Precision} = \frac{30}{130} = 76.9\%$$

- Out of the point we have predicted to be positive.

How many are correct?

* Recall →

	Diagnosed sick	Diagnosed Healthy
sick	1000	200
Healthy	800	900

- Out of the sick patients how many are they labeled correctly?

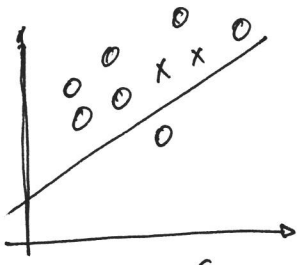
$$\text{Recall} = \frac{1000}{1200}$$

Critical field

	Level span	Level Tutor
span	100	170
Not span	30	700

$$\text{Recall} = \frac{100}{270} = 37\%$$

Critical field



$$\text{Recall} = \frac{6}{7}$$

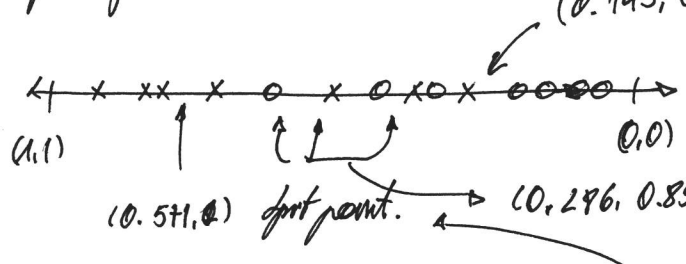
$$\text{Precision} = \frac{6}{8}$$

- Out of the points labeled positive, how many are predict correctly?
  ↳ specified in target.

Related to Recall.

* ROC curve →

-Receiver operating characteristic



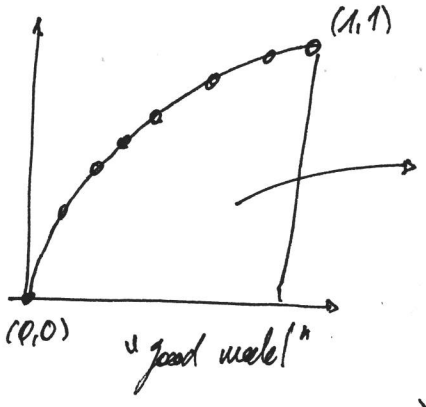
Repeat right → 1.0 ; all classified good.

Move boundary around and calculate

$$\text{True positive rate} = \frac{\text{True positives}}{\text{All positives.}}$$

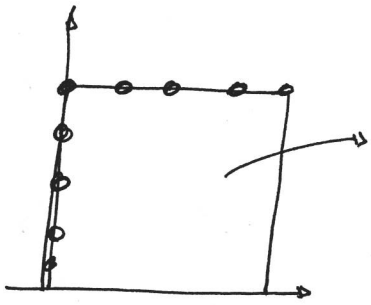
$$\text{False positive rate} = \frac{\text{false positive}}{\text{All negatives.}}$$

↳ vector (T pos, F positive)



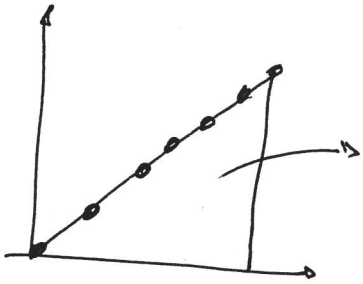
Area = 0.8

For perfect split



Area = 1.

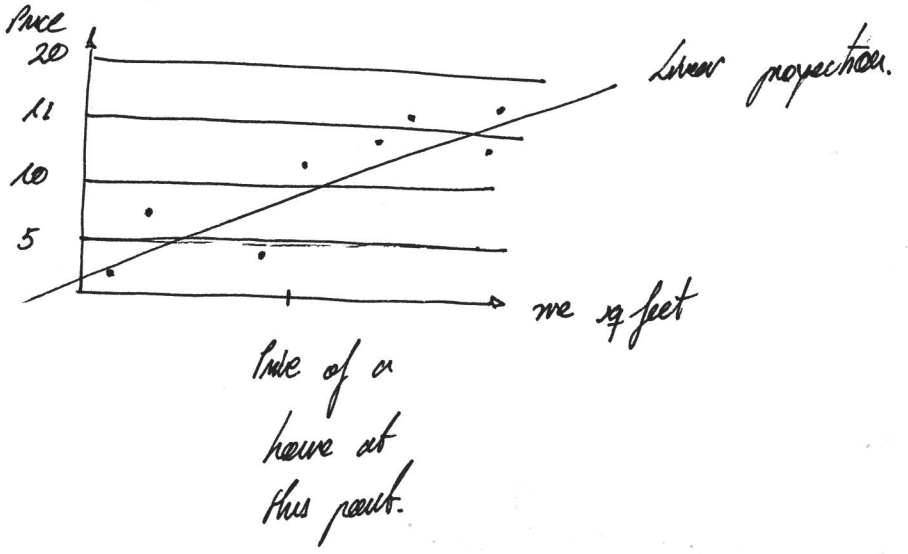
Random split



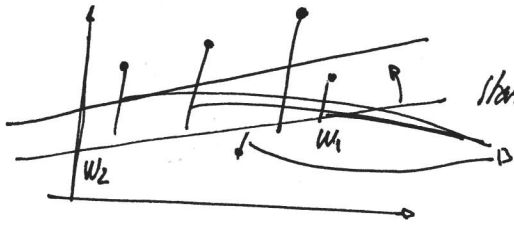
Area = 0.5

Linear Regression →

- House price example →



Rule of a house at this point.



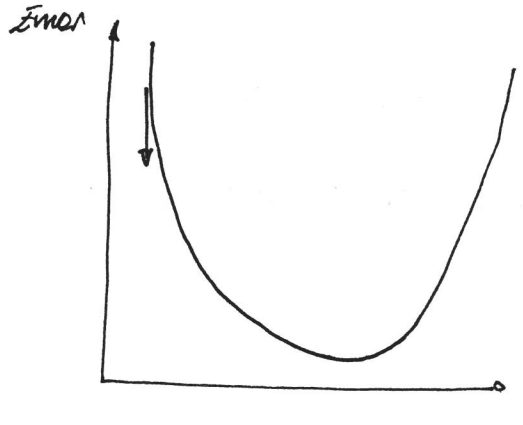
Start with random point.

Error ≡ measure of how far we are along the line.

$y = w_1 x + w_2$ * how the line to decrease error

* Point is to measure the error. that depends on the points and line.

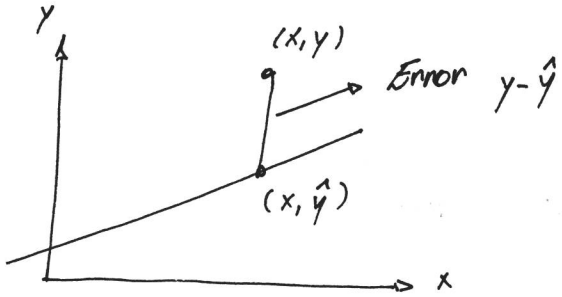
* the gradient descent to go down the error function until you reach the minimum.



learning rate.

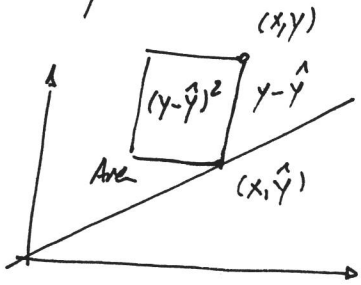
$w_i \longrightarrow w_i - \alpha \frac{\partial \text{Error}}{\partial w_i}$

* Mean absolute error ⟶



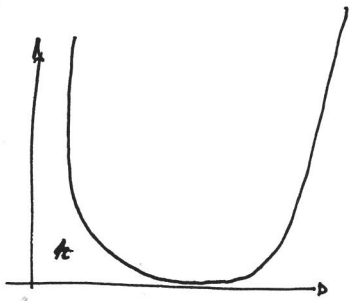
$$\text{Error} = \frac{1}{m}\sum_{i=1}^{m} |y-\hat{y}|$$

* Mean squared errors ⟶



$$\text{Error} = \frac{1}{2m}\sum_{i=1}^{m} (y-\hat{y})^2$$

* Penalizes more the error



function.

* Summed errors ⟶

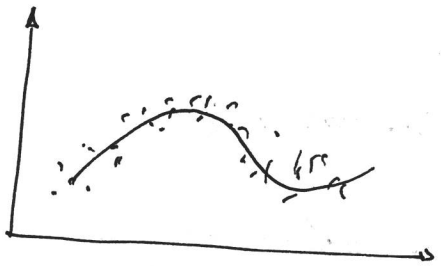
$$\text{Error} = \frac{1}{2}(y-\hat{y})^2 \longrightarrow \hat{y}(w_1, w_2) = w_1 x + w_2$$

$$\frac{\partial \text{Error}}{\partial w_1} = \frac{\partial \text{Error}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_1} = -(y-\hat{y}) \cancel{\#} x$$

$$\frac{\partial \text{Error}}{\partial w_2} = \frac{\partial \text{Error}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_2} = -(y-\hat{y})$$

* Total vs mean error ⟶ only difference between the two is the number of samples. In gradient descent you will multiply by α, learning rate anyway.

* Polinomial regression ⟶



$\hat{y} = w_1 x^3 + w_2 x^2 + w_3 x + w_4$ ⟶ Algorithm is the same.

the model line is what changes.

* Regularization ⟶

- Says box for the complexity on the model, adding weights to the error function.

- Simple models tend to generalize better.

- L1 regularization ⟶

$$\text{Error} = + \sum_i |w_i| + \text{Original error}$$

↑ absolute value.

- L2 regularization ⟶

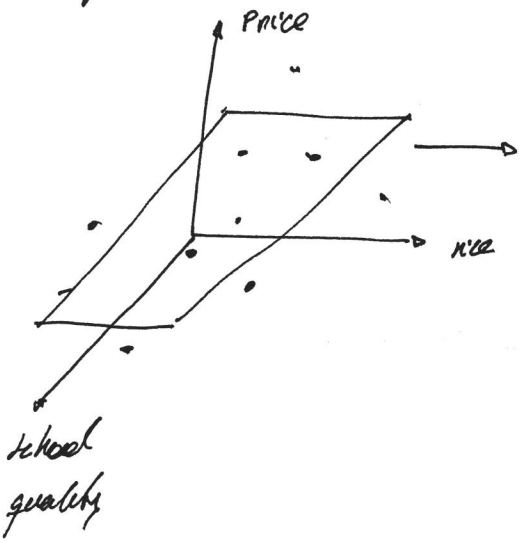
$$\text{Error} = + \sum_i (w_i)^2 + \text{original error}$$

- We use the parameter lambda to give some idea of the complexity of the model ⟶

* If lambda is small punishes the complex model less.

* On the contrary, with large λ simple models are achieved.

- Multiple linear regression →



$Price = w_1(size) + w_2(school) + w_3$

$x_1, x_2, \ldots, x_{n-1}$ → n dimensional space

prediction

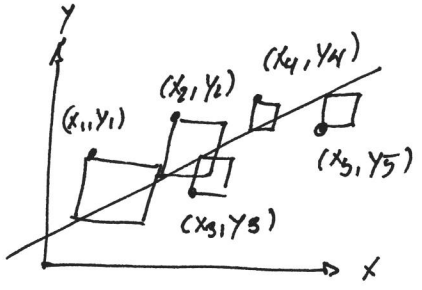
$\hat{y} = w_1 x_1 + \ldots + w_{n-1} x_{n-1} + w_n$

- Closed form solution →

$x_1, x_2, \ldots, x_m$

$y_1, y_2, \ldots, y_m$

→ $\hat{y}_i = \cancel{w_1 x_i + \ldots + w_2} = w_1 x_i + w_2$



$$E(w_1, w_2) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$

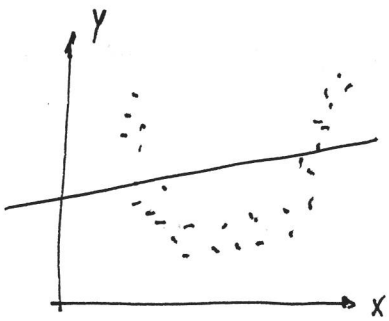
$$0 = \sum \frac{x_i^{\cancel{2}}}{m} w_1 + \sum \frac{\cancel{x_i}}{m} w_2 + \frac{\sum x_i y_i}{m}$$

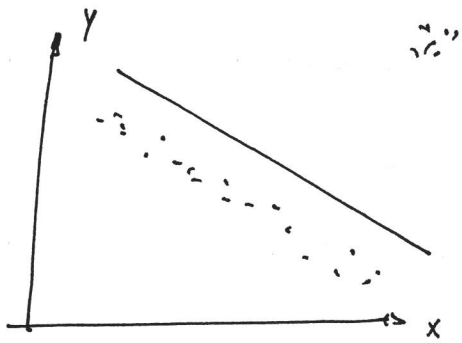
$$0 = \frac{\sum x_i}{m} w_1 + w_2 + \frac{\sum y_i}{m}$$

2 equations
2 unknown

very expensive, gradient descent is more effective.

* Linear regression has its limitations, it works if the data is linear →

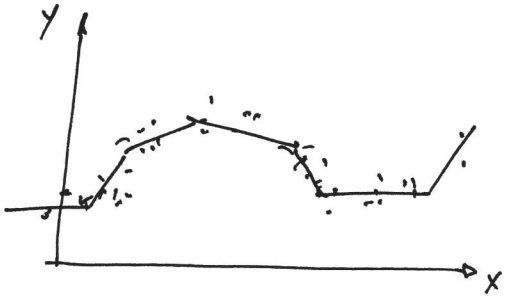




* It is sensitive to outliers →

L1 regularization	L2 regularization
Computationally inefficient (differential of the 1 / 1 abs).	Efficient
Sparse outputs Less performance	Non-sparse.
Feature selection	No feature selection.

* Neural Network →



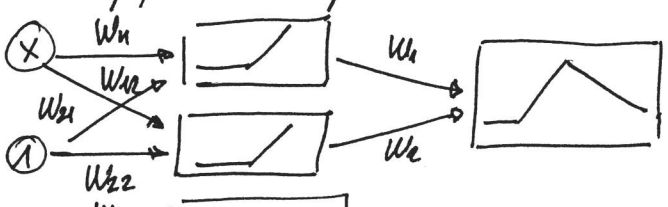
Instead of polynomial, combination of lines that fit our data well

- We can take out the ignored function since we don't need to do any classification. →

weighted sum of inputs to that layer

- use the mean squared error or another instead of cross-entropy (not classification).

Activation function ReLU



Ignored