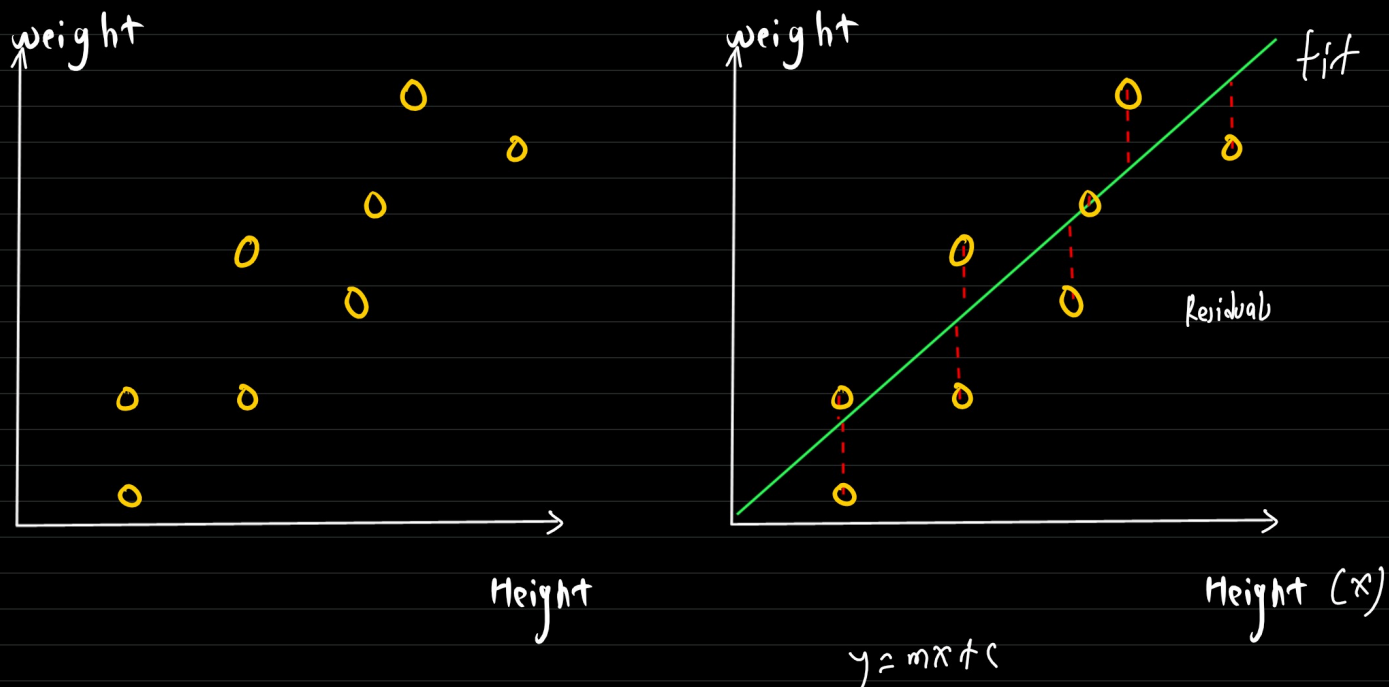
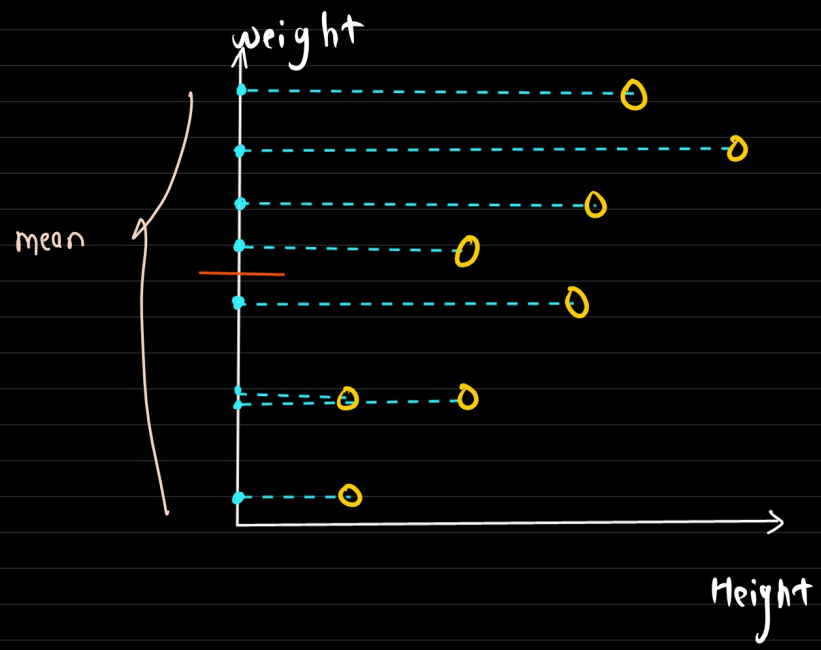
05-10-2025

Agenda:

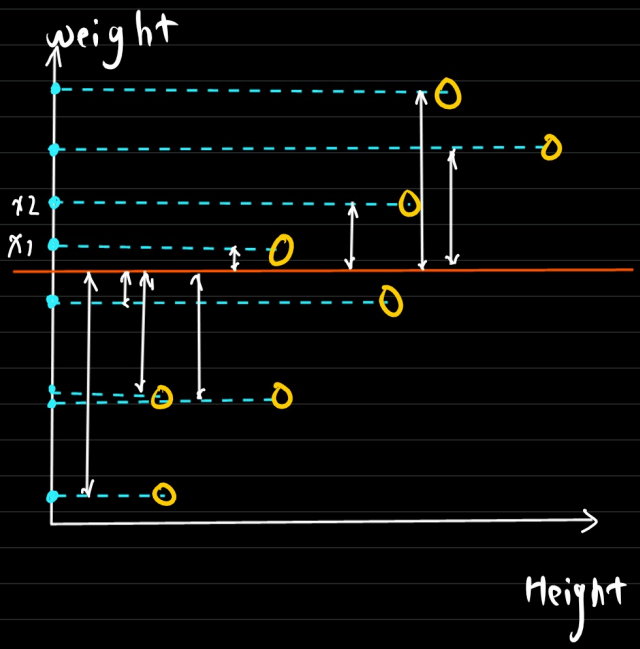
- → R^2 score
- → Bias & variance
- → L1 & L2 Regularization
- → practical
- → Logistic Regression (Depth)

R-squared score





$$\frac{\sum y_i}{m} =$$



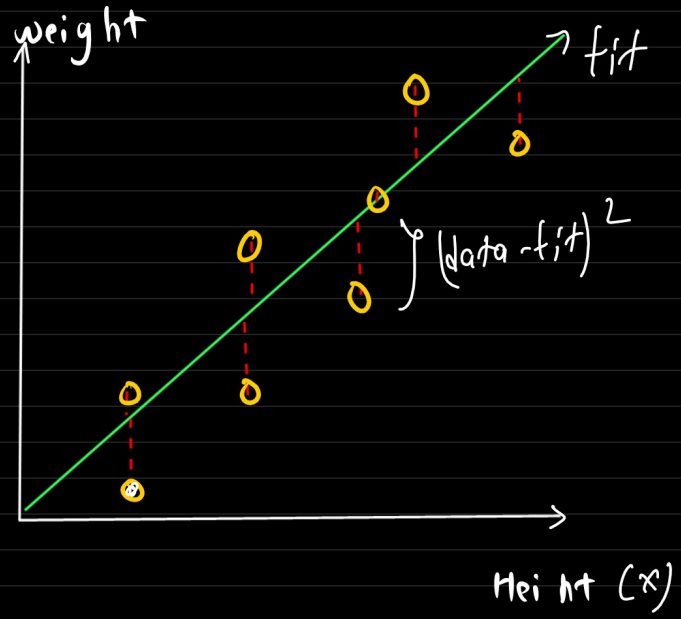
mean of sum of squares SS(mean)

$$= (\text{data} - \text{mean})^2$$

variation around mean =

$$\frac{(\text{data} - \text{mean})^2}{n}$$

$$= \frac{SS(\text{mean})}{n}$$



sum of squares around fit

SS(fit)

$$= (\text{data} - \text{fit})^2$$

$$R^2 = \frac{\text{var(mean)} - \text{var(fit)}}{\text{var(mean)}} \qquad \frac{\text{var(weight)} - \text{var(fit)}}{\text{var(weight)}}$$

how much of the variation in person weight we can explain by taking person height into account.

$$\text{var(mean)} = 72$$
$$\text{var(fit)} = 24$$

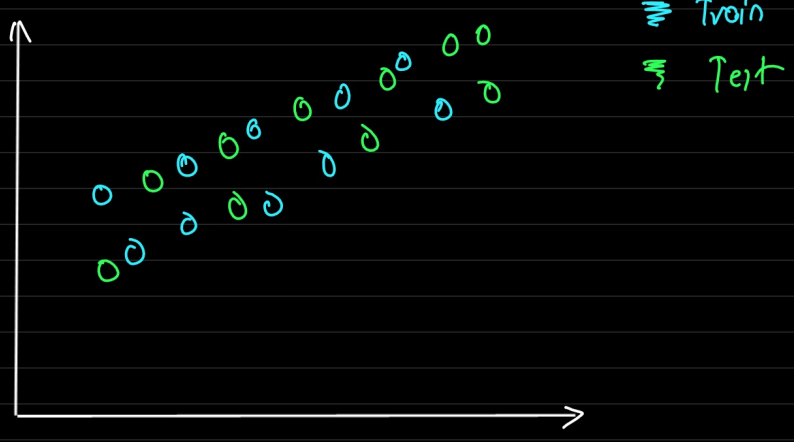
$$R^2 = \frac{72 - 24}{72} = 0.66 \approx 60\%.$$

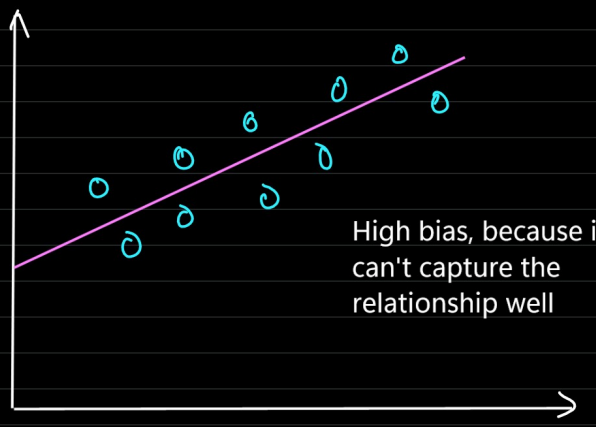
there is 60% reduction in variance when we take person height in account.

person height explains 60% of the variation in person weight.

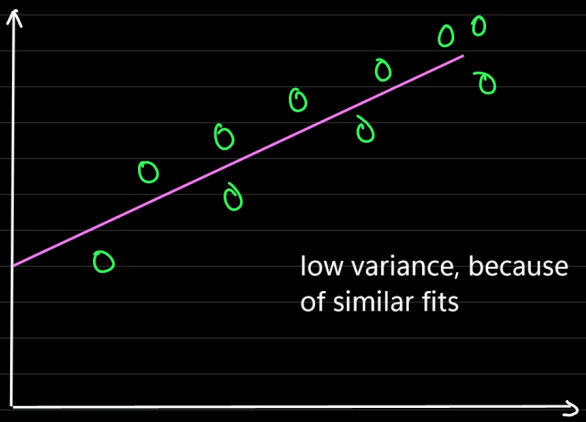
R^2 tells us how much of the variation in the target variable is explained by the model. Higher is better, but context matters.

Bias & Variance





High bias, because it can't capture the relationship well

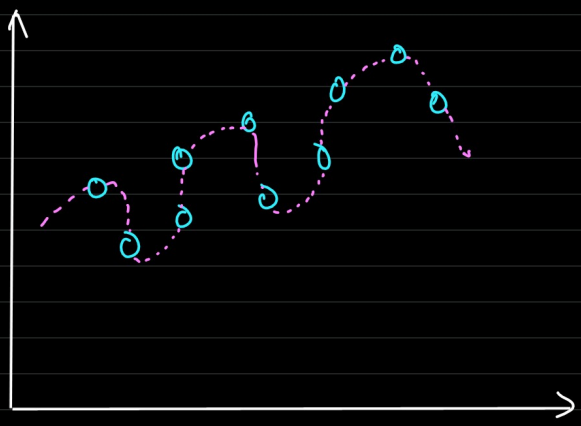


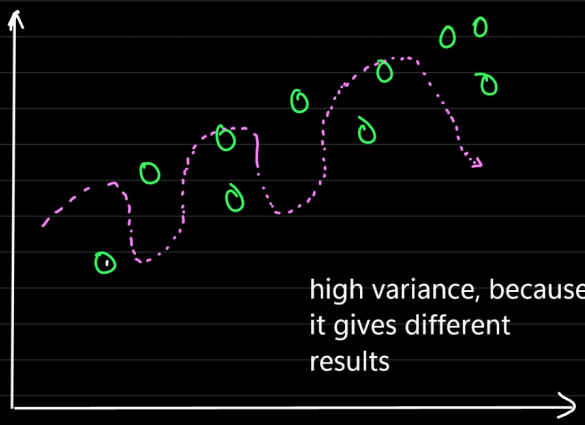
low variance, because of similar fits

loss (Train) : 108 ⟷ loss (Test) : 98

good fit

The inability for a ML model like linear regression to capture true relation is called as bias.





high variance, because it gives different results

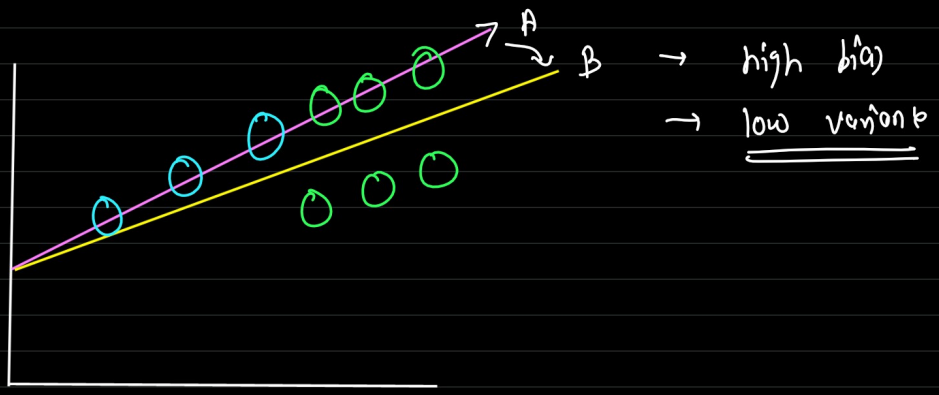
Low bias, because it is more flexible

loss (Train) : 0 ⟷ loss (Test) : 70

The difference in fits (train, test) is called as variance. The bigger the difference, the higher the variance.

→ Bias & variance tradeoff

→ low bias & low variance (ideal)



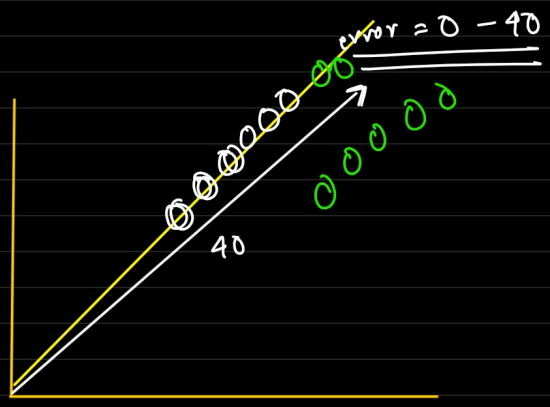
→ high bias
→ low variance

Loss (Train) = 0 : low bias
Loss (Test) = 100 : high variance

Regularization :

Driving Test

→ control, Drive ∞, ○, Traffic rijm
→ Tackling,



Don't just minimize error, keep the coefficients controlled.

The Problems with Vanilla Linear Regression

OLS/SGD works perfectly if:

- You have plenty of data compared to the number of features (columns).
- Features are not strongly correlated (no multicollinearity).
- The model is not too complex (not too many parameters).

Don't just minimize error, keep the coefficients controlled and to not overfit the training data.

L1 (Lasso):

$Loss_{new}$ = Modified MSE + $\lambda \sum |B_i|$

X → 11 features
m → 11 → coefficient / parameter → 100, 200, 300
c → 1

$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + c$

$y = M^T \cdot X + c$

$\beta \rightarrow \beta_1 + \beta_2 + \beta_3$

$$= \frac{1}{2m} \sum_{i=1}^{n} (y - \hat{y})^2 + \lambda \sum |B_i|$$

X → 11 features, 11 → $B_1 \rightarrow B_{11}$

loss = 400 + 353

loss ↓ = 753 + [ ]

$B_1 = 1$ 0
$B_2 = 200$
$B_3 = 150$
$B_4 = 2$

y.0 = loss → reduce coefficient value

$m_1 x_1$
0

: multicollinearity (feature correlated with another features)
: shrinks coefficient towards 0, but never 0

L2 (Ridge) : penalty : square of coefficients

Loss_{new} = Modified MSE + $\lambda \sum B_i^2$

: feature selection (picks only the most important feature)

: coefficient shrinks all the way to exact 0

Linear Regression (vanilla):

Loss (OLS): SSE → R^2 / F1

Loss (OLS): SSE + L1

SSE + $\lambda \sum |B_j|$ → R^2 / F1

Loss (OLS): SSE + L2

: SSE + $\lambda \sum (B_j)^2$ → R^2 / F1

Loss (GD): M-MSE → R^2 / F1

Loss (GD): M-MSE + L1

M-MSE + $\lambda \sum |B_j|$ → R^2 / F1

Loss (GD): M-MSE + L2

: M-MSE + $\lambda \sum (B_j)^2$ → R^2 / F1

feature → 100 → OLS

→ 1 million data points → GD

low - medium

iter ↙ low - high

→ R^2 score (Evaluation function / method)

→ Bias & variance tradeoff

→ Regularization

→ L1 → add penalty (sum of absolute values of coeff) to loss

→ L2 → add penalty (sum of square values of coeff) to loss

Logistic Regression

Used: Classification

BMI = height, weight

$y = x_1, x_2$

$\hat{y} = x_1 m_1 + x_2 m_2$

regression ← ∞ $\hat{y}$ = ↓

70	171
70.1	171.9
70.0001	188.8
∞	∞

Temp. ml → sick (yes/no) → classification.

∞	yes / no
↓	
$x_1 m_1$	$\hat{y}$ (yes / no)

$x_1, x_2 \rightarrow$ ML algo $(m_1 x_1 + m_2 x_2) \rightarrow \hat{y}$

↓

m_1, m_2

$x_1, x_2 \rightarrow$ ML algo $(m_1 x_1 + m_2 x_2) \rightarrow z \rightarrow$ function $\rightarrow \hat{y}$

↓

m_1, m_2

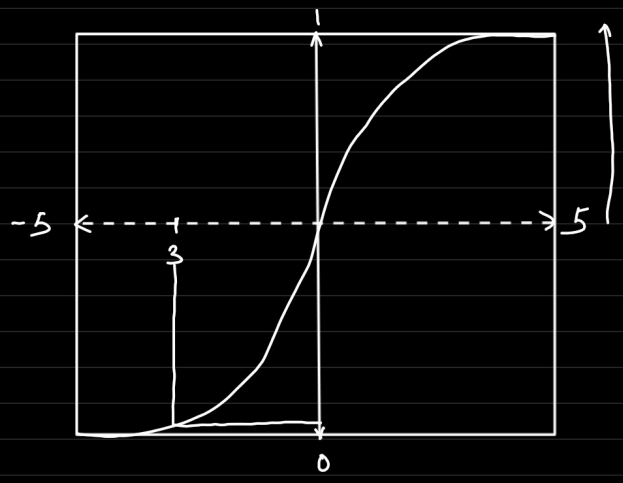
function → sigmoid → logistic function

$$\text{sigmoid} = \frac{1}{1 + e^{-z}}$$

logistic regression

- transformation of (z) into (p) btw 0-1
- The linear part $z = mx + c$

→ This is sigmoid or logistic function.



+ve → 1
-ve → 0

sigmoid (0 - 1)

sigmoid (-3) → 0.1

→ 0.0001, 0.04, 0.3, 0.8, 0.9, 1

Temp	y (if > 0.8)	$\hat{y}$
39.9	1	0.3 → 80% chance person is sick

sigmoid (x) > 0.5 → 1 else 0