kokchun giang

when your model is too complex, you can **regularize** it to make it simpler

many features leads to many parameters, which makes the model too complex

many features → model has many parameters

$$y = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

→ risk for multicollinearity

→ redundancy in features

→ inaccurate estimate of w_i

→ overfitting

$$\begin{pmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{pmatrix}$$

each column is a feature

want to ↑ bias
↓ variance

add a **penalty** to decrease the role of parameters other than the bias

ridge regression

(l_2-regularization)

↑λ => ↓variance

↑bias

↑

penalty parameter

$0 \leq \lambda \leq 1$

↑λ => w_i close to 0

$i \geq 1$

lasso regression

(l_1-regularisation)

↑λ => ↓variance

↑bias

$0 \leq \lambda \leq 1$

↑λ => least important features set to zero

elastic net

combines l_1 & l_2

Hyperparameters

λ - penalty

α - l_1-ratio

↑

$0 \leq \alpha \leq 1$

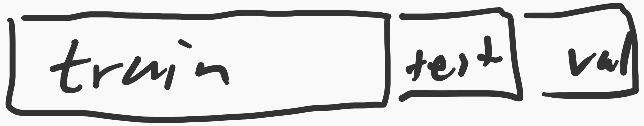
regularisation models require **feature scaling**

the models are trained using numerical approach such as gradient descent

=> require feature scaling

e.g.
- feature standardization
- normalization

k-fold cross validation for hyperparameter tuning



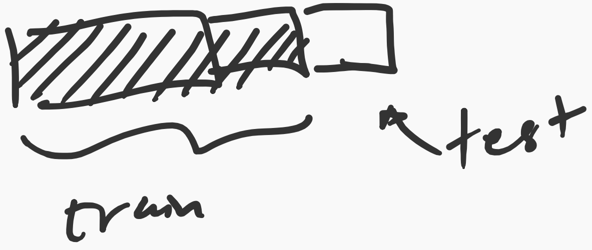
1. train w. different hyperparameters

2. predict & evaluate on validation data

3. choose new values on the hyperparameters & repeat 1. & 2.

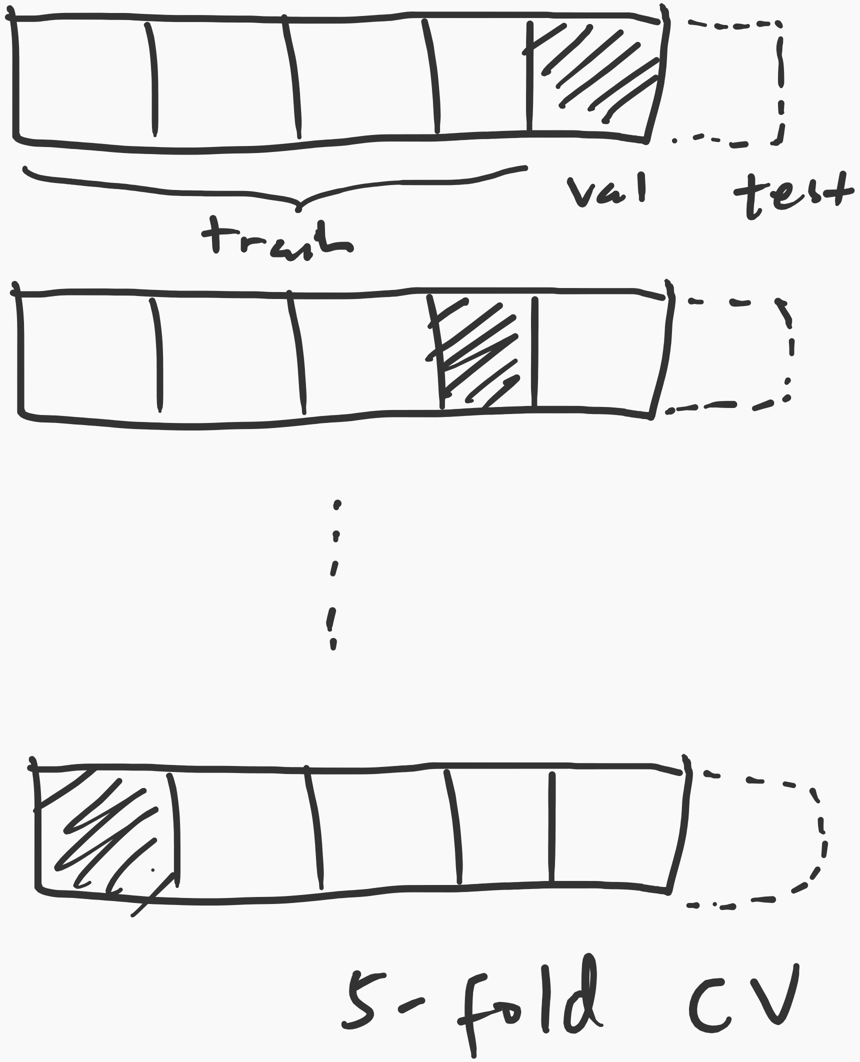
4. choose the value on hyperparam that gave least validation error

5. train on train & val datasets



6. evaluate on test set

k-fold cross validation for hyperparameter tuning



compute mean error

for small datasets, we utilises the data well

for larger it costs as we repeat training k times