naive bayes theorem

$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Using Bayes Theorem:

- $P(\text{Yes} \mid X) = \dfrac{P(X \mid \text{Yes})P(\text{Yes})}{P(X)}$ → 3/10
- $P(\text{No} \mid X) = \dfrac{P(X \mid \text{No})P(\text{No})}{P(X)}$ → 7/10

since they are same, no need to compute

$P(\text{Refund=No} \mid \text{Yes}) \times P(\text{Divorced} \mid \text{Yes}) \times P(\text{income}=120\text{k} \mid \text{Yes})$

continuous attributes

↳ discretization: partition into bins

↳ probability density estimation: assume normal distri. calculate mean and std dev → calculate probability

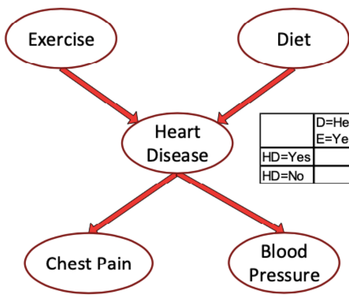
* if one probability is zero all becomes zero

↳ laplace estimate = $\dfrac{n_c+1}{n+v}$ v = total number of att val x_i can take (ex martial status = 3)

bayesian belief network =

Exercise=Yes	0.7
Exercise=No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	D=Healthy E=Yes	D=Healthy E=No	D=Unhealthy E=Yes	D=Unhealthy E=No
HD=Yes	0.25	0.45	0.55	0.75
HD=No	0.75	0.55	0.45	0.25

	HD=Yes	HD=No
CP=Yes	0.8	0.01
CP=No	0.2	0.99

	HD=Yes	HD=No
BP=High	0.85	0.2
BP=Low	0.15	0.8

$P(\text{HD} \mid \text{E=No}, \text{D=Yes}, \text{CP=Yes}, \text{BP=High})$

Yes = $P(\text{HD=Yes} \mid \text{E=No}, \text{D=Yes}) \times P(\text{Yes} \mid \text{CP=Yes}) \times P(\text{Yes} \mid \text{High})$

$0.45 \times 0.8 \times 0.85$

$= 0.374$

No = 0.0009 Yes > No → classification → Yes

artificial neural networks

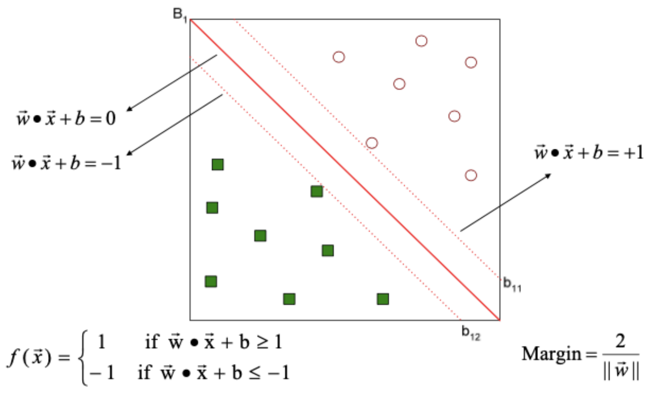
update rule for learning weights = $w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})\,x_{ij}$ (gradient descent in multilayer)

- if data is not linearly separable, perceptron not enough → multi layer

number of nodes in input / output layer =
- one for binary attribute / class
- k or $\underbrace{\log_2 k}_{\text{each combination for one of } k}$ nodes for k-class attribute

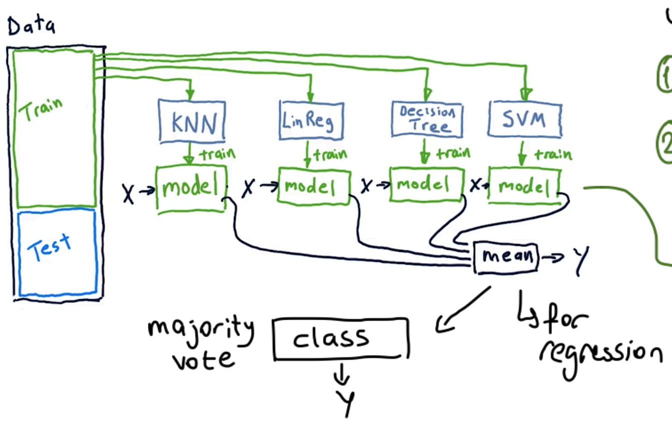
- if network is too large → overfitting

support vector machines



- if not linearly separable
  - ↳ use penalties in loss function
  - ↳ kernels → transformation to higher dimension space

ensemble methods



works better than a single classifier if =

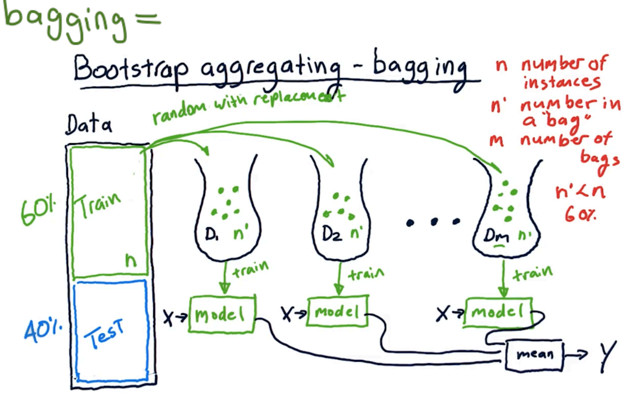
① all classifiers are independent

② all classifiers perform better than random guessing

↳ ex: error rate < 0.5 for binary classification

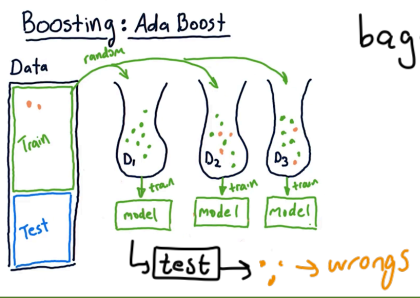
- these models can be same algorithms with different parameters

bagging =



→ building ensembe classes using the same algorithm but training each learner on a different set of the data

boosting =



bagging + train data for next model is weighted

↳ poorly classified in previous model has greater change to be chosen for next model to be trained