

week 5

the big picture

- every classifier = a rule that turns a row's features into a class
- learning = pick the rule from a family of rules that makes the fewest mistakes on train
- the real goal = few mistakes on rows we have not seen → generalization
- three choices define any model → model shape and loss and optimiser
- this week
 - a family of rules = boxes then trees
 - how to score a rule = confusion matrix and metrics
 - how a tree picks its rule = impurity → entropy or gini
 - how to stop it memorising = pruning
 - how to beat one tree = many trees → forest and boosting

classification

what

- supervised task that predicts a class label for a row
 - malignant or benign → classification cuz the answer is a label
 - tumour size in mm → regression cuz the answer is a number

formula

- $X = \{ (x_t, r_t) \}$ for $t = 1 \dots N$
 - $r = 1$ if positive sample
 - $r = 0$ if negative sample
- $x = x_1 x_2 \dots x_n$ = input variables = features = the columns
 - breast cancer table → radius mean and texture mean ... compactness mean
- $r = y$ = target = output variable
- N = number of rows and t = which row

read it

- positive = the class we're looking for and not the good one
 - malignant = positive and benign = negative
 - spam = positive and normal mail = negative
- flow → training set goes into the learning algorithm → it outputs a hypothesis $h(x) \rightarrow h(x)$ maps a new row's features to a target

vs

- regression predicts a number and its error is a distance $(y - \hat{y})$

- classification predicts a label and its error is right or wrong → so we count mistakes instead of measuring distance

kinds of task

- supervised = every training row has the right answer → learn the map from x to y
- unsupervised = no answers → find structure in x alone
- classification → supervised and the answer is one name from a fixed set
 - email spam or not → the answer is one of 2 names
 - credit rating bad or fair or good → still classification → the labels have an order but there are only 3 of them and nothing in between
- regression → supervised and the answer is a number on a scale
 - credit score 300 to 850 → any value in between makes sense → regression
 - house price → regression
- clustering → unsupervised → group rows that look alike with no names given
 - split customers into groups by spending → nobody told us the groups
- association → unsupervised → find things that happen together
 - people who buy bread also buy spread → a rule between items and not a prediction per row
- deciding test
 - is there a right answer in the data → no → clustering or association
 - yes → is it a name from a fixed list → classification
 - is it a number where 2.5 between 2 and 3 makes sense → regression
- borderline → star rating 1 to 5 → classification if each star is its own class and regression if 3.5 is a fine prediction

kinds of classification

- binary → 2 classes → malignant or benign → metrics use one positive class
- multiclass → more than 2 classes and one per row → Iris setosa or versicolor or virginica → metrics per class then averaged
- multilabel → a row can have several labels at once → a film is action and comedy → not on the slides

hypothesis

what

- hypothesis h = one guessed rule that splits positive from negative

- H = hypothesis class = every rule of that shape we could pick
- learning = search H for the best h

example

- slide $\rightarrow$ 2 features $\rightarrow$ texture mean x_2 and compactness mean x_5
($te_1 \leq \text{texture} \leq te_2$) AND ($co_1 \leq \text{compactness} \leq co_2$)
- $h(x) = 1$ if x is inside the box
- $h(x) = 0$ if x is outside
- H here = every axis aligned rectangle $\rightarrow$ 4 numbers te_1, te_2, co_1, co_2
pick one h



formula

- $E(h|X) = \sum 1(h(x_i) \neq r_i)$
- $1(\dots) = 1$ when the statement is true and 0 when false $\rightarrow$ so E counts the wrong rows
- 10 rows and the box misplaces 2 $\rightarrow E = 2$
- error rate = E / N
- accuracy = $1 - E / N$

why it works

- choosing H is an assumption about the world $\rightarrow$ we bet that malignant cells sit inside some range of both features
- if the true pattern is a box and H holds boxes $\rightarrow$ enough data finds it
- if the true pattern is a circle $\rightarrow$ no box fits perfectly $\rightarrow$ a small error stays no matter how much data
- bigger $H \rightarrow$ can fit more patterns but can also fit the noise $\rightarrow$ overfit risk

errors in the box

- C = the true class region and h = our box
- false positive = actually negative but h says positive
 - benign patient told malignant $\rightarrow$ extra tests and stress
- false negative = actually positive but h says negative
 - malignant patient sent home $\rightarrow$ the worse mistake here
- TP and TN = right
- S = most specific h = tightest box around all the + points
- G = most general h = biggest box that still has no - inside
- every box between S and G has zero training error $\rightarrow$ all fit train equally
- margin = gap between S and G

- pick the h halfway $\rightarrow$ a new point that lands a little off still gets the right side
- same reason a road is built in the middle of a gap and not touching a cliff $\rightarrow$ analogy
- CAREFUL the slide writes FP as $C = 1$ but $h = 0$ and FN as $C = 0$ but $h = 1$
- right way $\rightarrow$ FP = $C = 0$ and $h = 1$ and FN = $C = 1$ and $h = 0$

link to trees

- the slide box is a depth 4 decision tree
- texture $\geq te_1 \rightarrow$ texture $\leq te_2 \rightarrow$ compact $\geq co_1 \rightarrow$ compact $\leq co_2 \rightarrow$ positive
- so a tree = many boxes stuck together $\rightarrow$ every leaf is one box in feature space

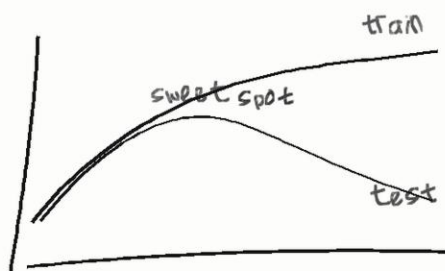
generalization and overfitting

what

- generalization = doing well on rows the model never saw
- overfitting = fits train including the noise $\rightarrow$ worse on new rows
- underfitting = too simple to catch the real pattern $\rightarrow$ bad everywhere

why it happens

- train error always falls as the model gets more complex $\rightarrow$ more knobs to bend around each row
- test error falls at first then rises $\rightarrow$ after the real pattern is caught the extra knobs only fit noise
- noise is different in new data $\rightarrow$ fitting it is pure loss



derive

- expected error on new data splits into three parts
error = bias² + variance + noise
- bias = how far the average model is from the truth $\rightarrow$ too simple = high bias
- variance = how much the model changes if you retrain on another sample $\rightarrow$ too complex = high variance
- noise = randomness in the labels $\rightarrow$ no model removes it

- one deep tree $\rightarrow$ bias low and variance high
- one split stump $\rightarrow$ bias high and variance low
- every fix this week moves along this trade off
 - pruning $\rightarrow$ less variance more bias
 - forest $\rightarrow$ less variance same bias
 - boosting $\rightarrow$ less bias

read it

train	test	means
1.00	0.72	overfit
0.60	0.68	underfit
0.85	0.83	good

knobs

- slide factors $\rightarrow$ complexity of the model and amount of training data and generalization error on new data
- more data with variety $\rightarrow$ harder to memorise $\rightarrow$ the gap closes
- duplicated rows add no information $\rightarrow$ they only count the same row twice

breaks when

- the test set is not like the future $\rightarrow$ iid broken $\rightarrow$ the test score lies

test set

- train error is too optimistic $\rightarrow$ the model chose θ to fit exactly those rows
- only rows the model never touched show how it does on new rows
- data leakage = any information from the test rows reaching training $\rightarrow$ the test score comes out too high $\rightarrow$ the model does worse when deployed
 - scaler fit on all rows before the split $\rightarrow$ test min and max leak in
 - a feature made from the target $\rightarrow$ eg days in hospital to predict admission
 - the same patient in train and test
- rule $\rightarrow$ split first $\rightarrow$ fit everything on train only $\rightarrow$ use test once at the end

three decisions for any model

what

- data $x = \{x_t, r_t\}$ is iid = independent and identically distributed
 - independent $\rightarrow$ one row does not affect another
 - identically distributed $\rightarrow$ train rows and future rows come from the same population

- this is the only reason training on old rows says anything about new ones

formula

- model $g(x|\theta) \rightarrow$ shape of the rule and θ = its parameters
- loss $L(r, g) \rightarrow$ cost of one prediction

$$E(\theta|X) = \sum L(r_t, g(x_t, \theta))$$
- optimiser $\rightarrow$ method that finds the best θ
 $\theta^* = \text{argmin over } \theta \text{ of } E(\theta|X)$
- argmin = the θ that gives the smallest E and not the smallest E itself

example

	model	loss	optimiser
linear reg	$\theta_0 + \theta_1 x$	Squared error	least squares
box	rectangle	0 or 1	pick between S and G
tree	tree of tests	impurity	greedy splits
adaboost	weighted stumps	exponential	add one stump at a time

why it works

- any new model = answer the three questions $\rightarrow$ then you know how it learns
- exam trick $\rightarrow$ describe a model by its three choices

confusion matrix

what

- table of counts of predicted vs actual
- slide layout $\rightarrow$ rows = predicted and columns = actual

	actual 1	actual 0
pred 1	TP	FP
pred 0	FN	TN

- CAREFUL sklearn `confusion_matrix` puts actual on rows and predicted on columns $\rightarrow$ the opposite way
- name rule $\rightarrow$ second letter = what we predicted and first letter = right or wrong
 - FN = false + negative $\rightarrow$ we said negative and were wrong $\rightarrow$ actually positive

example

- 100 patients and 10 have malignant cancer
- model flags 8 and 6 of them really are
- TP 6 and FP 2 and FN 4 and TN 88

- check $\rightarrow TP + FN = 10$ actual positives and $FP + TN = 90$ actual negatives

kinds

- card fraud model $\rightarrow$ positive = fraud
 - TP $\rightarrow$ fraud and flagged $\rightarrow$ money saved
 - TN $\rightarrow$ normal and passed $\rightarrow$ customer happy
 - FP $\rightarrow$ normal but blocked $\rightarrow$ customer annoyed and calls the bank
 - FN $\rightarrow$ fraud but passed $\rightarrow$ money lost

derive

- every metric below = one count divided by a row or a column of this table
- = a conditional probability like $P(Y|X)$ from week 1

accuracy

asks

- of all predictions what share is right

formula

$$\text{accuracy} = (TP + TN) / \text{total}$$
$$= (6 + 88) / 100 = 0.94$$

derive

$$\text{accuracy} = P(\text{predicted} = \text{actual})$$

breaks when

- one class is rare
- always say benign $\rightarrow (0 + 90) / 100 = 0.90$ and finds zero cancers
- rule $\rightarrow$ accuracy of always guessing the biggest class = share of that class $\rightarrow$ the model must beat this baseline
 - Orthopedic $\rightarrow$ biggest class $150 / 310 = 0.48 \rightarrow$ so 0.72 is a real gain

precision

asks

- when I say positive how often am I right

formula

$$\text{precision} = TP / (TP + FP)$$
$$= 6 / 8 = 0.75$$

derive

$$\text{precision} = \text{P}(\text{actual} + | \text{predicted} +)$$

- denominator = everything I flagged = the predicted positive row

read it

- 0.75 → 1 in 4 alarms is false
- matters when a false alarm is costly
 - spam filter → FP = a real mail sent to the spam folder → the user never sees a job offer or a bill → worse than one spam in the inbox
 - court verdict → FP = an innocent person jailed

recall or sensitivity

asks

- of the real positives how many did I catch

formula

$$\begin{aligned}\text{recall} &= \text{TP} / (\text{TP} + \text{FN}) \\ &= 6 / 10 = 0.60\end{aligned}$$

derive

$$\text{recall} = \text{P}(\text{predicted} + | \text{actual} +)$$

- denominator = every real positive = the actual positive column

read it

- 0.60 → 4 of 10 cancers missed
- matters when missing one is costly → cancer screening or fraud

VS

- recall goes up when you flag more → flag everyone → recall 1.0 and precision = base rate 0.10
- precision goes up when you flag less → flag only the surest one → precision maybe 1.0 and recall 0.1
- so neither alone means anything → look at both or at F1

specificity

asks

- of the real negatives how many did I clear

formula

$$\text{specificity} = \text{TN} / (\text{TN} + \text{FP})$$

$$= 88 / 90 = 0.98$$

derive

- specificity = $P(\text{predicted} = \text{actual}^-)$
- = recall of the negative class
- False positive rate = $FP / (FP + TN) = 1 - \text{specificity}$
- CAREFUL the slide writes $TN / (TN + FN) \rightarrow$ wrong cuz FN are actual positives $\rightarrow$ the denominator must be all actual negatives = $TN + FP$

F1

asks

- one number that is high only if precision and recall are both high

Formula

$$F1 = 2 \times P \times R / (P + R)$$
$$= 2 \times 0.75 \times 0.60 / 1.35 = 0.67$$

derive

- harmonic mean = 1 over the average of 1 over each
- $F1 = 2 / (1/P + 1/R)$
- plug in P and R as counts and it simplifies
- $F1 = 2TP / (2TP + FP + FN)$
- $= 12 / (12 + 2 + 4) = 0.67$
- TN never appears $\rightarrow$ F1 ignores the easy negatives $\rightarrow$ that is why it works for rare classes

why harmonic

- $1/P$ blows up when P is small $\rightarrow$ so the small one dominates
- P 1.0 and R 0.1 $\rightarrow$ normal mean 0.55 and F1 0.18

read it

- 0 to 1 and higher is better
- CAREFUL slide says higher F1 means higher accuracy $\rightarrow$ they are different numbers
- always benign model $\rightarrow$ accuracy 0.90 and F1 0

multiclass metrics

what

- K classes $\rightarrow$ each class takes a turn as positive and all the rest are negative $\rightarrow$ then average

example

- lab Iris matrix in sklearn layout $\rightarrow$ rows actual and columns predicted

actual	set	ver	vir
setosa	19	0	0
versicolor	0	12	0
virginica	0	1	13

$$\text{accuracy} = \text{diagonal} / \text{total} = 44 / 46 = 0.98$$

- sklearn layout $\rightarrow$ precision = diagonal / its column and recall = diagonal / its row
 - versicolor $P = 12 / 13 = 0.92$ and $R = 12 / 12 = 1.00$
 - virginica $P = 13 / 13 = 1.00$ and $R = 13 / 14 = 0.93$

formula

$$\text{macro} = (m_1 + m_2 + \dots + m_n) / K$$

$$\text{weighted} = \sum (\text{support of class} / N) \times m$$

- support = actual rows of that class in the test set $\rightarrow 19, 12, 14$

$$\text{macro } P = (1.00 + 0.92 + 1.00) / 3 = 0.97$$

read it

- macro $\rightarrow$ every class counts the same $\rightarrow$ shows if a small class is failing
- weighted $\rightarrow$ big classes count more $\rightarrow$ sits near accuracy
- macro much below weighted $\rightarrow$ a small class is bad
 - Orthopedic $\rightarrow$ Hernia recall 0.33 $\rightarrow$ macro recall 0.66 vs weighted 0.72
- micro $\rightarrow$ add up TP and FP and FN over all classes first $\rightarrow$ then compute once
 - Iris $\rightarrow$ TP 44 and FP 1 and FN 1
 - micro $P = 44 / (44 + 1) = 0.98$
- in single label multiclass every wrong row is one FP for the class it got and one FN for its real class $\rightarrow$ so micro $P = \text{micro } R = \text{micro } F1 = \text{accuracy}$
- use micro when every row should count the same and macro when every class should count the same
- binary $\rightarrow$ report the positive class itself and not an average
- multilabel $\rightarrow$ micro and accuracy differ cuz a row can be half right

log loss

asks

- how much probability did the model put on the right answer

formula

- $\log \text{loss} = -(1/n) \sum [y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$
- $\hat{y}$ = predicted probability of class 1 and $\log$ = natural log
- only one term is alive per row
 - $y = 1 \rightarrow \text{cost} = -\log(\hat{y})$
 - $y = 0 \rightarrow \text{cost} = -\log(1 - \hat{y})$

derive

- the model says $P(y=1) = \hat{y} \rightarrow$ so the probability it gave to what really happened is
 - $P(y) = \hat{y}^y \times (1 - \hat{y})^{(1-y)}$
 - $y = 1 \rightarrow \hat{y}$ and $y = 0 \rightarrow 1 - \hat{y}$
- rows are iid $\rightarrow$ the chance of the whole dataset = the product over rows = likelihood
- best model = the one that makes the real labels most likely
- products of small numbers are awkward $\rightarrow$ take the log $\rightarrow$ product becomes sum
- maximise the log likelihood = minimise its negative $\rightarrow$ divide by n for a mean $\rightarrow$ log loss

example

y	$\hat{y}$	cost
1	0.9	0.11
1	0.5	0.69
1	0.1	2.30
1	0.01	4.61

- 3 rows $y = 1$ 0 1 and $\hat{y} = 0.9$ 0.2 0.6
- likelihood = $0.9 \times 0.8 \times 0.6 = 0.432$
- log loss = $-\log(0.432) / 3 = 0.28$

read it

- 0 = perfect and no top $\rightarrow$ lower is better
- 0.69 = what a coin flip scores on 2 classes $\rightarrow$ beat that
- confident and wrong costs the most $\rightarrow$ the model is punished for being sure

breaks when

- an unpruned tree gives pure leaves $\rightarrow \hat{y} = 1.0$ or $0.0 \rightarrow$ one wrong row costs $\log(0) = \infty \rightarrow$ sklearn clips $\hat{y}$ a bit away from 0
- CAREFUL slide formula has $(1 - y)(1 - \hat{y})$ without a log $\rightarrow$ needs $\log(1 - \hat{y})$
- CAREFUL slide says $0 \leq \log \text{loss} \leq 1 \rightarrow$ really 0 to ∞

kinds of metric

- label based $\rightarrow$ needs hard predictions $\rightarrow$ accuracy precision recall specificity F1 $\rightarrow$ uses predict

- probability based $\rightarrow$ needs the probabilities $\rightarrow$ log loss $\rightarrow$ uses predict_proba
- two models with the same accuracy can have very different log loss
 $\rightarrow$ the one that was sure when right and unsure when wrong wins

entropy

what

- entropy = how mixed a group is = average surprise of its labels

derive

- surprise of an event with probability p
 $\text{surprise} = -\log_2(p)$
- why log $\rightarrow$ two independent events multiply their p but surprise should add $\rightarrow$ only log turns product into sum
- why base 2 $\rightarrow$ a fair coin flip = exactly 1 bit
 - $p = 1 \rightarrow 0$ bits $\rightarrow$ no surprise
 - $p = 1/8 \rightarrow 3$ bits
- entropy = expected surprise = each surprise weighted by how often it happens
 $\text{Info}(D) = -\sum p \log_2(p)$ over each class

example

4 buy $\rightarrow -\log_2(4/7) = 0.81$ bits each

3 no $\rightarrow -\log_2(3/7) = 1.22$ bits each

$$\text{Info}(D) = (4/7)(0.81) + (3/7)(1.22) = 0.985$$

read it

- 0 = pure $\rightarrow$ all one class
- max = $\log_2(K)$ when every class is equal $\rightarrow$ 2 classes 1.0 and 3 classes 1.585
- why max at equal shares $\rightarrow$ you can not do better than a random guess there $\rightarrow$ every outcome is equally surprising
- any imbalance makes one class easier to guess $\rightarrow$ average surprise drops
 - 50 50 $\rightarrow$ 1.0 and 90 10 $\rightarrow$ 0.47 and 100 0 $\rightarrow$ 0
- calculator $\rightarrow \log_2(x) = \log(x) / \log(2)$
- $0 \times \log(0)$ counts as 0

information gain

what

- how much a split lowers entropy $\rightarrow$ ID3 picks the attribute with the highest gain

formula

$$\text{Info}_A(D) = \sum (|D_j| / |D|) * \text{Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

- D_j = the rows that go down branch j and $|D_j|$ = how many

derive

- after the split a random row lands in branch j with probability $|D_j| / |D|$
- so Info_A = the expected entropy left after asking about A
- gain = entropy before minus expected entropy after = what asking A tells you about the class
- gain is never negative $\rightarrow$ splitting can not make you know less on average
- gain = 0 when every branch keeps the parent mix
 - 14 rows 8 yes and 6 no split into 4 3 and 4 3 $\rightarrow$ each child 0.985 $\rightarrow$ gain 0 $\rightarrow$ the split is useless

example

- the 7 customers

id	age	student	credit	buy
1	young	yes	excellent	yes
2	young	no	fair	no
3	middle	no	excellent	yes
4	senior	no	excellent	yes
5	senior	no	fair	no
6	senior	no	fair	no
7	middle	no	fair	yes
8	middle	yes	excellent	?

count first

- all $\rightarrow$ 4 buy and 3 no
- young 1 1 and middle 2 0 and senior 1 2
- student yes 1 0 and student no 3 3
- excellent 3 0 and fair 1 3

$$\text{Info}(D) = 0.985$$

$$\text{Info}_{\text{age}} = (2/7)(1.0) + (2/7)(0) + (3/7)(0.918) = 0.679$$

$$\text{Gain}(\text{age}) = 0.985 - 0.679 = 0.31$$

$$\text{Info}_{\text{student}} = (1/7)(0) + (6/7)(1.0) = 0.857$$

$$\text{Gain}(\text{student}) = 0.13$$

$$\text{Info}_{\text{credit}} = (3/7)(0) + (4/7)(0.811) = 0.464$$

$$\text{Gain}(\text{credit}) = 0.52$$

- credit is not on the slides and it wins $\rightarrow$ credit = root

entropy lookup

counts	entropy	gini
1 1	1.000	0.500
1 2	0.918	0.444
1 3	0.811	0.375
2 3	0.911	0.480
1 4	0.722	0.320
3 4	0.986	0.490
2 6	0.863	0.408
5 9	0.940	0.459
4 0	0	0

breaks when

- attribute has many values $\rightarrow$ biased toward it $\rightarrow$ see gain ratio

gain ratio

what

CA5 fix for the many values bias of information gain

why it is needed

- split on customer id $\rightarrow$ 7 groups of 1 $\rightarrow$ all pure $\rightarrow$ Info = 0 $\rightarrow$ gain = 0.986 = the max possible
- but the id of a new customer matches no branch $\rightarrow$ the split predicts nothing
- more branches = smaller groups = purer by chance $\rightarrow$ gain rewards Chopping

formula

$$\text{SplitInfo_A(D)} = -\sum (|D_j|/|D|) \log_2(|D_j|/|D|)$$

$$\text{GainRatio(A)} = \text{Gain(A)} / \text{SplitInfo_A(D)}$$

derive

- SplitInfo = entropy of the branch sizes themselves and ignores the class
- = how much information you get just from knowing which branch a row took
- many equal small branches $\rightarrow$ SplitInfo up to $\log_2(v)$ $\rightarrow$ divides the gain down

example

$$\text{SplitInfo(age)} = \text{entropy of sizes } 2 \ 2 \ 3 = 1.557$$

$$\text{GainRatio(age)} = 0.31 / 1.557 = 0.20$$

- slide prints 1.52 $\rightarrow$ real 1.557 $\rightarrow$ ratio still 0.20

id $\rightarrow$ SplitInfo = $\log_2(7) = 2.81 \rightarrow$ ratio = $0.986 / 2.81 = 0.35$

credit $\rightarrow$ SplitInfo = entropy of 3 4 = $0.985 \rightarrow$ ratio = $0.52 / 0.985 = 0.53$

- credit still wins and id is pushed down

breaks when

- a branch is tiny $\rightarrow$ SplitInfo near 0 $\rightarrow$ ratio blows up $\rightarrow$ C4.5 only uses the ratio among attributes with at least average gain

gini index

what

- CART impurity measure $\rightarrow$ pick the split with the lowest weighted gini

formula

$Gini(D) = 1 - \sum p^2$ over each class

$Gini_A(D) = (|D_1|/|D|) Gini(D_1) + (|D_2|/|D|) Gini(D_2)$

reduction = $Gini(D) - Gini_A(D)$

derive

- pick 2 rows at random with replacement

- chance both are class i = $p \times p \rightarrow$ chance they match = $\sum p^2$

- chance they differ = $1 - \sum p^2 = gini$

- same as $\sum p(1-p)$ = chance you label a row wrong if you guess labels at random in the class shares

4 yes 3 no $\rightarrow 2 \times (4/7)(3/7) = 24/49 = 0.49$

read it

- 0 = pure

max = $1 - 1/K$ when classes are equal $\rightarrow$ 2 classes 0.5 and 3 classes 0.667

- CART only splits in 2 $\rightarrow$ an attribute with 3 values has 3 ways to make 2 groups $\rightarrow$ try all

nominal with v values $\rightarrow 2^{(v-1)} - 1$ two group options

$v = 3 \rightarrow 3$ and $v = 4 \rightarrow 7$

- why binary only

- any multi way split can be built from several binary ones $\rightarrow$ no power is lost

- multi way splits chop the rows into small groups fast $\rightarrow$ the many values bias of gain

- the same code then works for numbers $\rightarrow$ one threshold = one binary split

example

$$\text{Gini}(D) = 1 - (4/7)^2 - (3/7)^2 = 0.49$$

$$\text{young vs rest} = (2/7)(0.5) + (5/7)(0.48) = 0.486$$

$$\text{senior vs rest} = (3/7)(0.444) + (4/7)(0.375) = 0.406$$

$$\text{middle vs rest} = (2/7)(0) + (5/7)(0.48) = 0.343$$

$$\text{student} = (6/7)(0.5) = 0.429$$

$$\text{credit} = (4/7)(0.375) = 0.214$$

- credit lowest $\rightarrow$ wins again
- best age split = middle vs rest $\rightarrow$ reduction $0.49 - 0.343 = 0.15$
- CAREFUL slide says $\text{Gini}(D) = 0.51$ and young vs rest = 0.62 $\rightarrow$ right values 0.49 and 0.486

vs

- gini vs entropy $\rightarrow$ same shape and pick the same split almost always $\rightarrow$ gini has no log so it is faster
- why not just count the error (1 - biggest p) $\rightarrow$ it misses progress
 - 400 400 split into 300 100 and 100 300 $\rightarrow$ error 0.25
 - 400 400 split into 200 400 and 200 0 $\rightarrow$ error 0.25 too
- gini says 0.375 vs 0.333 $\rightarrow$ prefers the second cuz it made a pure node
- entropy says 0.811 vs 0.689 $\rightarrow$ same pick

decision tree

what

- decision tree induction = learning a tree from rows that all have labels
- root = first test and internal node = a test on one attribute
- branch = one outcome of the test and leaf = a class
- predict = start at the root and follow the matching branch to a leaf

why it works

- each test cuts the rows into groups that are purer than before
- keep cutting until each group is one class $\rightarrow$ the path to a leaf is an if then rule
- impurity is the score that says which cut helps most
 - like 20 questions $\rightarrow$ ask the question that halves the possibilities $\rightarrow$ analogy

history

- ID3 = Iterative Dichotomiser $\rightarrow$ information gain $\rightarrow$ multiway splits
- C4.5 = successor of ID3 $\rightarrow$ gain ratio $\rightarrow$ also handles numbers and missing values
- CART = Classification and Regression Tree $\rightarrow$ gini $\rightarrow$ binary splits $\rightarrow$ also regression

- sklearn DecisionTreeClassifier = CART → default criterion gini

kinds of attribute

- nominal → names with no order → colour red green blue or student yes no
 - split → one branch per value in ID3 or any 2 group option in CART
- ordinal → names with an order → age young middle senior or credit fair excellent
 - split → keep the order → young vs middle and senior is fine and young and senior vs middle breaks the order
 - v values → v - 1 order keeping cuts
- numeric → numbers → petal width
 - split → threshold → petal width ≤ 0.80
- borderline → zip code is digits but nominal → 60111 is not more than 60110 in any useful way
- customer id → nominal with every value different → gain loves it and it predicts nothing → drop it

learning process

- start with partition D and the attribute list and a selection method
- make node N for D
- all rows one class → N becomes a leaf of that class → stop
- else → score every attribute with the method → take the best splitting criterion
- label N with that test and grow one branch per outcome
- repeat on each branch with only its rows → recursion
- stop when a stopping rule hits and return the tree
- Stopping rules
 - node is pure
 - no attributes left → leaf = majority class
 - branch got no rows → leaf = majority class of the parent
 - a pre pruning limit → max depth or min rows

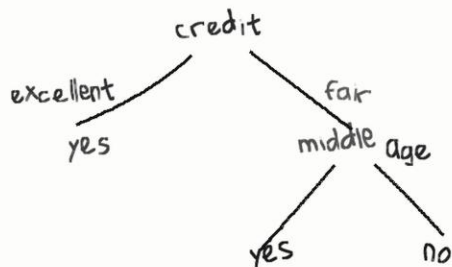
derive

- trying every possible tree is far too many → the tree is greedy → best split now and never undo
- fast but the final tree may not be the best tree
- numbers → sort the column and try a cut between each pair of neighbours → n values give n - 1 cuts
 - If is all 150 rows → root = petal width ≤ 0.80 → setosa leaf gini 0 and the rest gini 0.5
- root gini = $1 - 3 \times (1/3)^2 = 0.667$
- after = $(50/150)(0) + (100/150)(0.5) = 0.333$

- only the order of values matters $\rightarrow$ scaling never changes a tree

example

- the 7 customers $\rightarrow$ credit at root
- excellent $\rightarrow$ rows 1 3 4 $\rightarrow$ all yes $\rightarrow$ leaf yes
- fair $\rightarrow$ rows 2 5 6 7 $\rightarrow$ 1 yes 3 no $\rightarrow$ split again by age
 - middle $\rightarrow$ row 7 $\rightarrow$ yes and the rest $\rightarrow$ no $\rightarrow$ pure $\rightarrow$ stop



- customer 8 $\rightarrow$ excellent $\rightarrow$ buy yes
- the slide tree with age at the root comes from a bigger textbook table and not from these 7 rows

read it

- leaf probability = class shares in the leaf $\rightarrow$ leaf with 8 yes 2 no $\rightarrow$
 $P(\text{yes}) = 0.8 \rightarrow \text{predict_proba}$
- depth $d \rightarrow$ at most 2^d leaves
- feature importance = total impurity drop that feature caused weighted by rows reaching each node $\rightarrow$ scaled to sum to 1
 - Iris tree $\rightarrow$ petal width 0.92 $\rightarrow$ nearly the whole tree

knobs

- criterion $\rightarrow$ gini or entropy
- max_depth and min_samples_split and min_samples_leaf $\rightarrow$ pre pruning
- ccp_alpha $\rightarrow$ post pruning

breaks when

- no limit $\rightarrow$ grows until every leaf is pure $\rightarrow$ memorises noise $\rightarrow$ overfit
- unstable $\rightarrow$ change a few rows and the root can change $\rightarrow$ whole tree changes
- cuts are along one feature at a time $\rightarrow$ a diagonal border needs a staircase of many cuts
- greedy $\rightarrow$ can miss a pair of splits that only work together

regression trees

- CART also predicts numbers $\rightarrow$ DecisionTreeRegressor
- impurity = variance of y in the node $\rightarrow$ split = the one with the biggest drop in squared error
- a leaf predicts the mean y of its rows
 - leaf with prices 10 12 14 $\rightarrow$ predicts 12

- prediction is a staircase $\rightarrow$ flat inside each leaf

good at

- readable rules
- no scaling and handles numbers and categories
- fast predict $\rightarrow$ only depth many tests

tree pruning

what

- removing the least reliable branches $\rightarrow$ smaller tree $\rightarrow$ easier to read and better on new rows

why it is needed

- deep branches are built from few rows $\rightarrow$ a leaf made from 1 odd row is fitting noise
- fewer rows behind a decision = less trustworthy decision

pre pruning

- stop early $\rightarrow$ a test says the split is not worth it $\rightarrow$ node becomes a leaf
- the leaf holds the majority class or the class probabilities
- tests $\rightarrow$ gain below a threshold or depth limit or too few rows
- cheap but can stop too early $\rightarrow$ a weak split may lead to a strong one below it
- XOR $\rightarrow$ each feature alone gives gain 0 but both together split perfectly $\rightarrow$ pre pruning stops at the root

post pruning

- grow the full tree $\rightarrow$ then cut subtrees from the bottom up
- a cut subtree becomes a leaf labelled with the majority class of its rows
- keep the cut if error on held out data does not get worse

formula

- sklearn cost complexity pruning
- $R_\alpha(T) = R(T) + \alpha \times (\text{number of leaves})$
- $R(T)$ = training error or impurity of tree T
- α = price per leaf $\rightarrow$ bigger α $\rightarrow$ fewer leaves

derive

- same idea as week 2 Ridge $\rightarrow$ fit term plus a penalty for complexity
- $\alpha = 0 \rightarrow$ full tree and α huge $\rightarrow$ one leaf

- pick α with cross validation

VS

- pre = cheap and risks stopping too early
- post = costs more and sees the whole tree first so usually better
- both trade a bit more bias for much less variance

ensembles

what

- many models vote $\rightarrow$ the group beats one model
- lab uses them $\rightarrow$ the week 6 slides are not up yet

why it works

- if each voter is right more than half the time and their mistakes differ $\rightarrow$ the majority is right more often
- 3 voters at 0.7 each $\rightarrow P(2 \text{ or } 3 \text{ right}) = 0.784$
- 5 voters $\rightarrow 0.837$
- 25 voters $\rightarrow 0.98$
- breaks if voters make the same mistakes $\rightarrow$ 3 copies of one tree = one tree
- so the whole game is making the members different

kinds

- bagging $\rightarrow$ same model on different bootstrap samples in parallel $\rightarrow$ equal vote $\rightarrow$ random forest
- boosting $\rightarrow$ models in sequence each fixing the last ones mistakes $\rightarrow$ weighted vote $\rightarrow$ adaboost and gradient boosting
- voting or stacking $\rightarrow$ different model types vote $\rightarrow$ tree and SVM and logistic regression $\rightarrow$ not in the lab

random forest

what

- many deep trees each trained on a different random view of the data $\rightarrow$ majority vote
- = bagging = bootstrap aggregating + random features

formula

- bootstrap = draw N rows with replacement from N rows
- $P(\text{a row is never drawn}) = (1 - 1/N)^N \rightarrow 0.368$
- each tree sees about 63% of the distinct rows $\rightarrow$ the other 37% = out of bag rows = a free test set

- at each split only a random subset of features is tried $\rightarrow$ default $\sqrt{p}$ (features) for classification

derive

- averaging B models each with variance v and correlation c between them

$$\text{variance} = c \times v + (1 - c) \times v / B$$
- more trees shrink the second part only
- c stays $\rightarrow$ random features lower $c \rightarrow$ trees disagree more $\rightarrow$ the vote cancels more noise
 $c = 0.5$ and $B = 100 \rightarrow 0.505v$
 $c = 0.2$ and $B = 100 \rightarrow 0.208v$

read it

- bias about the same as one deep tree and variance much lower
- why bias stays $\rightarrow$ every tree is built the same way from the same kind of sample $\rightarrow$ their average prediction = one tree's average prediction $\rightarrow$ averaging moves nothing closer to the truth $\rightarrow$ it only removes the wobble
- why adaboost lowers bias $\rightarrow$ each stump adds a new cut where the current vote is wrong $\rightarrow$ the sum of stumps can draw shapes one stump can not
- more trees never overfit more $\rightarrow$ just slower
- lose the single readable tree $\rightarrow$ use `feature_importances_` instead

knobs

- `n_estimators` $\rightarrow$ 100 default and `max_features` and `max_depth`

example

- Orthopedic lab $\rightarrow$ tree 0.72 and forest 0.82 $\rightarrow$ 5 fold CV tree 0.81 and forest 0.86

adaboost

What

- trains small trees one after another $\rightarrow$ each new one focuses on rows the earlier ones got wrong $\rightarrow$ weighted vote
- base learner = stump = tree with 1 split $\rightarrow$ weak = barely better than guessing

formula

- start $\rightarrow$ every row weight $w = 1/N$
each round

err = sum of weights of wrong rows

$$\alpha = 0.5 \times \ln(1 - \text{err}) / \text{err}$$

wrong rows $\rightarrow w \times e^{\alpha}$

right rows $\rightarrow w \times e^{-\alpha}$

- then divide all w by their sum so they add to 1

- final

$$H(x) = \text{sign}(\sum \alpha \times h(x))$$

derive

- $\alpha > 0$ only when $\text{err} < 0.5 \rightarrow$ better than guessing gets a say

- err small $\rightarrow \alpha$ big $\rightarrow$ accurate stumps vote louder

$\text{err} = 0.5 \rightarrow \alpha = 0 \rightarrow$ a coin flip gets no say

- after the update the wrong rows hold exactly half the total weight
 $\rightarrow$ the previous stump now scores $\text{err} = 0.5 \rightarrow$ the next stump is forced to be different

example

- 10 rows and the stump misses 3

$$\text{err} = 3 \times 0.1 = 0.3$$

$$\alpha = 0.5 \times \ln(0.7 / 0.3) = 0.42$$

$$\text{wrong} \rightarrow 0.1 \times 1.53 = 0.153$$

$$\text{right} \rightarrow 0.1 \times 0.66 = 0.066$$

$$\text{total} = 0.458 + 0.458 = 0.917$$

after dividing $\rightarrow$ wrong 0.167 each and right 0.071 each

- each wrong row now weighs over 2x a right row

read it

- lowers bias $\rightarrow$ many weak rules add up to a strong one

- sklearn multiclass version = SAMME adds $\ln(K - 1)$ to $\alpha \rightarrow$ so a class guess better than $1/K$ counts

breaks when

- noisy labels or outliers $\rightarrow$ they keep getting wrong $\rightarrow$ their weight explodes $\rightarrow$ the model chases noise

knobs

- `n_estimators` $\rightarrow$ 50 default and `learning_rate` $\rightarrow$ shrinks each α

vs

	forest	adaboost
trees	parallel	in sequence
each tree	deep	stump
rows	bootstrap	reweighted
vote	equal	by α
lowers	variance	bias

exam check

- confusion matrix given $\rightarrow$ find TP FP FN TN $\rightarrow$ check which axis is actual $\rightarrow$ plug in
- which metric $\rightarrow$ name the costly mistake
 - disease or fraud $\rightarrow$ FN costly $\rightarrow$ recall
 - spam or verdict $\rightarrow$ FP costly $\rightarrow$ precision
 - rare class $\rightarrow$ F1 or recall and not accuracy
- derive a metric $\rightarrow$ write it as $P(\text{something} | \text{something})$
- compute entropy or gain $\rightarrow$ counts per branch first $\rightarrow$ lookup table $\rightarrow$ weighted sum
- gain ratio $\rightarrow$ why $\rightarrow$ many valued attributes
- gini $\rightarrow$ binary splits $\rightarrow$ list every 2 group option $\rightarrow$ pick lowest
- build a tree by hand $\rightarrow$ best root $\rightarrow$ repeat on each impure branch $\rightarrow$ stop when pure
- predict a new row $\rightarrow$ walk it down the tree
- overfit or underfit $\rightarrow$ train vs test
- pre vs post pruning $\rightarrow$ stop early vs cut after
- log loss $\rightarrow$ where it comes from $\rightarrow$ likelihood $\rightarrow$ why lower is better
- forest vs boosting $\rightarrow$ parallel and variance vs sequence and bias
- describe any model $\rightarrow$ model and loss and optimiser
- mix ups
 - precision = predicted positive side and recall = actual positive side
 - specificity = recall of the negative class
 - gain $\rightarrow$ highest and gini $\rightarrow$ lowest
 - ID3 gain and C4.5 gain ratio and CART gini
 - FP = false alarm and FN = missed case
 - sklearn rows = actual and slide rows = predicted

conceptual questions

- classification vs regression $\rightarrow$ a name from a fixed set vs a number on a scale $\rightarrow$ spam vs house price
- credit rating bad fair good $\rightarrow$ classification cuz 3 names with nothing between $\rightarrow$ credit score 300 to 850 $\rightarrow$ regression
- task kinds $\rightarrow$ classification and regression need answers in the data

- clustering and association do not → see kinds of task
- binary vs multiclass → 2 classes vs more than 2 with one per row → multiclass metrics are per class then averaged
- hypothesis → one rule h → hypothesis class H → every rule of that shape → H is the assumption and if the truth is not in H some error always stays
- generalization → doing well on unseen rows → train error can not show it cuz θ was chosen to fit those rows
- overfit → train high and test low → too complex or too little data → underfit → both low → too simple
- more data helps cuz new variety is harder to memorise → duplicates add no new information
- iid → rows independent and from the same population as future rows → without it train says nothing about test
- FP → said positive and wrong → FN → said negative and wrong → cancer FN worse cuz a sick patient goes home → spam FP worse cuz a real mail is hidden
- S and G → tightest and biggest box with zero train error → margin = gap → pick the middle so small noise does not flip the label
- accuracy lies on rare classes → 10 cancers in 100 → always benign = 0.90 accuracy and 0 recall
- precision → of my alarms how many real → recall → of the real ones how many caught → flag more = recall up precision down
- $F1$ harmonic → the smaller of P and R dominates → $P 1.0 R 0.1$ → mean 0.55 but $F1 0.18$
- specificity → $TN / (TN + FP)$ → recall of the negative class → $FPR = 1 - \text{specificity}$
- multiclass averages → macro = each class equal → weighted = by support → micro = pool counts = accuracy in single label
- log loss → scores the probability not just right or wrong → $-\log$ of the probability given to the truth → from maximum likelihood → confident and wrong → huge
- three decisions → model and loss and optimiser → linear reg $\theta_0 + \theta_1 x$ and squared error and least squares → tree and impurity and greedy splits
- tree building → pure → leaf → else best split → branch per outcome → recurse → stop at pure or no attributes or no rows or a limit
- entropy → average surprise → 0 when pure cuz no surprise → max at equal shares cuz guessing is hardest
- information gain → entropy before minus weighted entropy after → never negative → 0 when each branch keeps the parent mix
- gain ratio → gain loves many valued attributes like id → divide by split info → many small branches get pushed down
- gini → chance two random rows differ → same picks as entropy nearly always → faster with no log

- not plain error rate $\rightarrow$ it misses progress like making a pure node $\rightarrow$ 400 400 example
- CART binary $\rightarrow$ any multi way split = several binary ones $\rightarrow$ avoids chopping $\rightarrow$ 3 values $\rightarrow$ 3 options and v values $\rightarrow 2^{(v-1)} - 1$
- ID3 gain multi way $\rightarrow$ C4.5 gain ratio and numbers $\rightarrow$ CART gini binary and regression
- numeric $\rightarrow$ sort and try a threshold between neighbours $\rightarrow$ scaling keeps the order so the tree is the same
- attribute kinds $\rightarrow$ nominal any grouping $\rightarrow$ ordinal keep the order $\rightarrow$ numeric threshold
- greedy $\rightarrow$ best split now and never undo $\rightarrow$ fast but maybe not the best tree $\rightarrow$ XOR
- trees overfit $\rightarrow$ grow until pure $\rightarrow$ leaves from 1 row fit noise $\rightarrow$ pruning cuts the unreliable branches
- pre vs post pruning $\rightarrow$ stop early and cheap but may stop too soon like XOR vs grow then cut and costlier but better
- leaf $\rightarrow$ majority class $\rightarrow$ probability = class shares $\rightarrow$ 8 yes 2 no $\rightarrow$ yes with 0.8
- tree good $\rightarrow$ readable and no scaling and fast $\rightarrow$ bad $\rightarrow$ overfits and unstable and staircase borders and greedy
- ensemble $\rightarrow$ many voters with different mistakes $\rightarrow$ 3 at 0.7 $\rightarrow$ 0.784 $\rightarrow$ useless if all make the same mistake
- forest diversity $\rightarrow$ bootstrap rows and random features per split $\rightarrow$ lowers correlation so the vote cancels more noise
- bagging vs boosting $\rightarrow$ parallel equal vote lowers variance vs sequence weighted vote lowers bias
- adaboost $\rightarrow$ wrong rows $\times e^{\alpha}$ and right $\times e^{-\alpha}$ $\rightarrow$ vote weight $\alpha = 0.5 \ln((1 - \text{err}) / \text{err})$
- forest bias $\rightarrow$ same expected prediction as one tree so averaging keeps the bias $\rightarrow$ stumps add up to shapes one stump can not draw so boosting lowers bias
- support $\rightarrow$ actual rows per class $\rightarrow$ macro far below weighted $\rightarrow$ a small class is failing
- test set $\rightarrow$ kept apart so the score is honest $\rightarrow$ leakage = test info in training $\rightarrow$ score too high $\rightarrow$ lab scales leaks but trees do not care
- random_state $\rightarrow$ fixes the split and the trees random choices $\rightarrow$ without it Iris gave 1.00 or 0.978
- sklearn matrix $\rightarrow$ rows actual and columns predicted $\rightarrow$ precision = diagonal / column and recall = diagonal / row
- root by hand $\rightarrow$ count classes per branch $\rightarrow$ entropy or gini per branch $\rightarrow$ weighted sum $\rightarrow$ best gain or lowest gini
- feature importance $\rightarrow$ total impurity drop from splits on that feature weighted by rows $\rightarrow$ sum to 1

- regression tree $\rightarrow$ variance or squared error replaces impurity $\rightarrow$ leaf predicts the mean

lab code

- Iris (150 flowers and 4 measurements and 3 species) then Orthopedic (310 patients and 6 spine angles and 3 classes)
- every model $\rightarrow$ make it then fit then predict then score

imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- 1 numpy $\rightarrow$ arrays and maths
- 2 pandas $\rightarrow$ tables = DataFrame
- 3 matplotlib.pyplot $\rightarrow$ plots
- 4 seaborn $\rightarrow$ nicer statistical plots on top of matplotlib

load

```
url = "https://raw.githubusercontent.com/
eiflin/datasets/main/Iris.csv"
data = pd.read_csv(url)
data = data.drop(columns='Id')
data.head()
```

- 1 pd.read_csv $\rightarrow$ reads the csv from the link into a table
- 2 drop(columns='Id') $\rightarrow$ removes the row number column $\rightarrow$ an id is not a feature and would only help memorise rows
- 3 head $\rightarrow$ shows the first 5 rows

explore

```
data.info()
data.describe()
sns.countplot(data["Species"], palette="Set2")
plt.title("Species")
plt.show()
```

- 1 info $\rightarrow$ column names and non null counts and dtypes $\rightarrow$ 150 rows and no missing values and Species is object = text
 - 2 describe $\rightarrow$ count and mean and std and min and quartiles and max per numeric column
 - 3 countplot $\rightarrow$ bar of how many rows per class $\rightarrow$ Iris is 50 50 50 so balanced
 - 4 title and show $\rightarrow$ label the plot and draw it
- why check the class counts $\rightarrow$ an unbalanced target makes accuracy

misleading

- Orthopedic → Spondylolisthesis 160 and Normal 100 and Hernia 60 → not balanced

```
color_list = ['red' if i=='Iris-setosa' else  
              'blue' if i=='Iris-versicolor' else  
              'green'  
              for i in data.loc[:, 'Species']]
```

```
pd.plotting.scatter_matrix(  
    data.loc[:, data.columns != 'Species'],  
    c=color_list, figsize=[15, 15], diagonal='hist')  
sns.pairplot(data=data, hue='Species')
```

1 color_list → one colour per row picked from its species

2 data.loc[:, data.columns != 'Species'] → every row and every column except Species

3 scatter_matrix → a grid of scatter plots of every feature against every other and histograms on the diagonal

4 pairplot → the same grid in seaborn coloured by hue

- what to read → petal length and petal/width split setosa off clearly
→ expect a very accurate tree

preprocess

```
X, y = data.loc[:, data.columns != 'Species'], data.loc[:, 'Species']  
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()
```

```
X_scaled = pd.DataFrame(scaler.fit_transform(X))
```

1 bc → X = the 4 feature columns and y = the Species column

2 MinMaxScaler → a tool that squeezes each column into 0 to 1

$$x' = (x - \min) / (\max - \min)$$

3 fit_transform → learns each columns min and max then applies the formula

4 pd.DataFrame → wraps the array back into a table → column names are lost and become 0 1 2 3

split

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(  
    X_scaled, y, test_size = 0.3, random_state = 32)
```

1 train_test_split → shuffles and splits the rows

2 test_size = 0.3 → 30% test → Iris 105 train and 45 test and Orthopedic 217 and 93

3 random_state = 32 → seed so the same split comes out every run

decision tree

```
from sklearn import tree
```

```
dtree = tree.DecisionTreeClassifier()
```

```
dtree.fit(X_train, y_train)
```

```
prediction = dtree.predict(X_test)
```

```
print('Prediction: {}'.format(prediction))
```

```
print('Decision tree accuracy is: ', dtree.score(X_test, y_test))
```

1 DecisionTreeClassifier → empty CART tree → defaults criterion gini and no depth limit → so no pruning

2 fit → grows the tree on the training rows using the split search above

3 predict → runs each test row down the tree → returns a species per row

4 format → puts the array into the printed text

5 score → accuracy on the test rows = predict then compare with y_test

- lab Iris → 0.978 → 44 of 45

- lab orthopedic → 0.72

plot the tree

```
plt.figure(figsize=(15,15))
```

```
tree.plot_tree(dtree)
```

```
plt.show()
```

```
import graphviz
```

```
dot_data = tree.export_graphviz(dtree, out_file=None,
```

```
    feature_names=X.columns.values.tolist(),
```

```
    class_names=y.unique(),
```

```
    filled=True, rounded=True, special_characters=True)
```

```
graph = graphviz.Source(dot_data)
```

```
graph
```

1 figure → a 15 by 15 inch canvas so the tree fits

2 plot_tree → draws every node with its test and gini and samples and value

3 export_graphviz → writes the tree as DOT text → out_file=None returns the text instead of saving a file

4 feature_names → real column names since X_scaled lost them

5 class_names → names for the classes in the boxes

6 filled → colours each node by its majority class and rounded → round boxes

7 graphviz.Source → renders the DOT text as a picture

- reading a node → samples = rows reaching it and value = rows per class and gini = impurity and class = majority

- CAREFUL class_names=y.unique() lists classes in order of first appearance but sklearn orders them alphabetically → dtree.classes_

- Iris → both orders are the same by luck

- Orthopedic → unique gives Hernia Spondylolisthesis Normal and

sklearn uses Hernia Normal Spondylolisthesis → Normal and Spondylolisthesis are swapped in the picture
fix → `class_names=dtree.classes_`

confusion matrix and report

```
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(y_test, prediction)
sns.heatmap(cf_matrix, annot=True)
from sklearn.metrics import classification_report
print(classification_report(y_test, prediction))
```

1 confusion_matrix → counts with rows = actual and columns = predicted → classes in alphabetical order

2 heatmap → colours the matrix and `annot=True` writes the numbers in the cells

3 classification_report → precision and recall and F1 and support per class plus accuracy and macro avg and weighted avg

- lab Iris → only 1 mistake → one virginica called versicolor

- lab Orthopedic tree matrix → order Hernia Normal Spondylolisthesis

`[[ 7, 14, 0],`

`[ 7, 26, 1],`

`[ 0, 4, 34]]`

- Hernia recall $7 / 21 = 0.33$ → 14 hernias called Normal → those two classes overlap

- Spondylolisthesis F1 0.93 → easy class

- accuracy 0.72 and macro F1 0.67 and weighted F1 0.71

random forest

```
from sklearn.ensemble import RandomForestClassifier
randforest = RandomForestClassifier()
randforest.fit(X_train, y_train)
prediction = randforest.predict(X_test)
randforest.score(X_test, y_test)
```

`tree.export_graphviz(randforest.estimators_[1], ...)`

1 RandomForestClassifier → 100 trees by default each on a bootstrap sample

2 fit and predict and score → same calls as the tree → predict is the majority vote

3 estimators_ → the list of fitted trees → [1] picks the second one to draw cuz all 100 cannot be shown

- lab → Iris 0.978 and Orthopedic 0.82

adaboost

```
from sklearn.ensemble import AdaBoostClassifier
adaBoost = AdaBoostClassifier(n_estimators=100)
```

```
adaBoost.fit(X_train, y_train)
```

```
prediction = adaBoost.predict(X_test)
```

```
adaBoost.score(X_test, y_test)
```

- 1 AdaBoostClassifier → boosted stumps → $n_estimators=100$ = up to 100 rounds → default 50
- 2 fit → trains stumps one by one reweighting the missed rows
- 3 predict → weighted vote of the stumps
- lab → Iris 1.00 and Orthopedic 0.78 → default 50 rounds there

exercise 2

- same calls on the Orthopedic file with the target column `class_status`
- the notebook keeps the outputs but the code cells are empty → fill them by copying exercise 1 and changing url and Species → `class_status`
- CAREFUL the tree and forest have no `random_state` → each run gives a slightly different score
 - inside one tree sklearn shuffles the feature order at every node → when two splits score the same the first one tried wins → a different shuffle can pick a different split
 - inside a forest the bootstrap rows and the random features also change
- Iris tree on the same split gave 1.00 or 0.978 across reruns
- CAREFUL scaling does nothing for trees → same split order → over 50 splits the Orthopedic tree averages 0.788 with or without `MinMaxScaler`
- CAREFUL the scaler is fit on all rows before the split → test rows leak into the min and max → harmless for trees but wrong for scaled models
- CAREFUL the 0.72 is one split → over 50 random splits the Orthopedic tree ranges 0.70 to 0.88
- CAREFUL `countplot` with palette and no hue gives a FutureWarning → write `sns.countplot(x="Species", hue="Species", data=data)`

doing it properly

- not on the slides
- split with `stratify=y` so each class keeps its share in train and test `train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)`
- set `random_state` on the model too `tree.DecisionTreeClassifier(random_state=0)`
- skip the scaler for trees and forests
- prune by searching the knobs with cross validation on train only `GridSearchCV(tree.DecisionTreeClassifier(random_state=0), {'max_depth': [2, 3, 4, 5, None]}, cv=5)`

- Orthopedic 5 fold CV $\rightarrow$ no limit 0.81 and depth 3 0.82 and depth 4 0.83 and depth 2 0.77
- forest 0.86 and adaboost 0.80 by the same CV $\rightarrow$ the forest wins
- compare models by CV mean and not by one test split
- unbalanced classes $\rightarrow$ report macro F1 and not only accuracy
- `class_weight='balanced'` $\rightarrow$ makes missed small classes like Hernia count more