CHARTA LEARNING · LESSON NOTES

Text data and training a model

Quantitative Methods

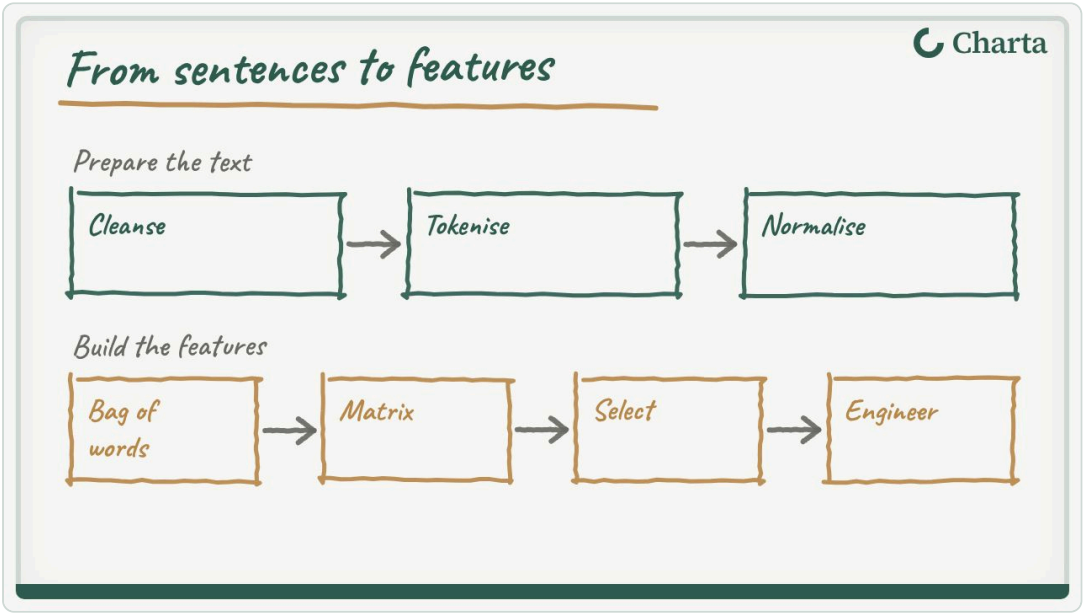
1 A pile of words, a model that wants numbers

- Case: Kestrel predicts credit downgrades from calls
- 2,000 transcripts, each labelled downgrade or not
- A model needs numbers; transcripts are words
- Step 1: text into features. Step 2: train the model

IN THE LESSON

- Meet Dana Whitfield, an analyst at a made up firm called Kestrel Credit Research.
- Kestrel wants to predict which companies will have their credit rating cut, using what managers say on earnings calls.
- Dana has two thousand call transcripts, and each one is labelled by whether a downgrade followed.
- Here is her problem.
- A model is a machine that does arithmetic, and sentences are not numbers.
- Think of a librarian who must describe every book using only a tally chart.
- First she must decide what counts as a word, then which words matter.
- This lesson follows Dana through those choices, and then through training her model.

2 From sentences to features



IN THE LESSON

- Here is the whole journey on one board, so you always know where we are.
- Read the top row first.
- Dana cleanses the text, which means stripping out junk such as web page code.
- She tokenises it, which means cutting it into small units called tokens, usually words.
- She normalises it, which means making different forms of one word match.
- Now the bottom row.
- She counts the tokens in a bag of words and stacks the counts into a matrix, which is just a table of numbers.
- Then she selects the tokens worth keeping, using frequency measures.
- Last, she engineers extra features, new numbers built from the text.
- We start with the ideas in the bottom row, and then come back to cleaning in more detail.

3 Tokens, and making them match

- Token: the unit a model counts, usually a word
- Normalise: lower case, drop stop words
- Stemming: crude chop, downgraded to downgrad
- Lemmatisation: real dictionary word, slower

IN THE LESSON

- A token is the basic unit of text that a model counts.
- Usually it is a word, though it can be a number or a mark of punctuation.
- Tokenisation is the job of cutting text into tokens.
- Now a puzzle.
- The words downgrade, downgraded and downgrading mean nearly the same thing, but a computer sees three different strings.
- Normalisation fixes that.
- First, put everything in lower case, so Profit and profit match.
- Second, remove stop words, the little function words such as the, of and a, which carry almost no meaning.
- Third, shrink each word to a common form.
- Stemming chops off the ending crudely, giving downgrad.
- Lemmatisation is gentler and maps to a real dictionary word, downgrade, but costs more computing effort.

4 Example 1: one sentence, normalised

We do not expect a downgrade as leverage falls.

Tokenise	cut at the spaces	**9 tokens**
Drop stop words	we, do, a, as removed	**5 tokens**
Stem	downgrade to downgrad	**5 tokens**
Result	not expect downgrad leverag fall	**5 tokens**

IN THE LESSON

- Example one.
- Here is one sentence from a call: we do not expect a downgrade as leverage falls.
- Step one, tokenise by cutting at the spaces.
- Count the words: we, do, not, expect, a, downgrade, as, leverage, falls.
- That is nine tokens.
- Step two, put everything in lower case and remove the stop words we, do, a and as.
- We deliberately keep the word not, and you will see why shortly.
- Four removed from nine leaves five.
- Step three, stem.
- Downgrade becomes downgrad, leverage becomes leverag, and falls becomes fall.
- Still five tokens, now in a form that will match other sentences.
- The result is not, expect, downgrad, leverag, fall.

5 Counting: the bag of words

- Bag of words: which tokens occur, and how often
- Order of the words is lost
- Document term matrix: one row per document
- One column per token; cells are counts

IN THE LESSON

- Now we count.
- A bag of words is exactly what it sounds like.
- Tip all the tokens of a sentence into a bag, shake it, and note which tokens are inside and how many times each appears.
- The order is lost.
- Dana does this for every sentence and stacks the results into a document term matrix.
- Here is the rule you must not forget.
- Each row is one document, which here means one sentence.
- Each column is one token.
- Each cell holds how many times that token appears in that sentence.
- With four sentences and six hundred different tokens, the table is four rows by six hundred columns.
- Swapping rows and columns is the classic slip, so say it aloud: rows are documents, columns are tokens.

6 Example 2: a document term matrix

Rows are sentences, columns are tokens

Sentence A	revenue 1, downgrad 0, weak 0	**5 tokens**
Sentence B	revenue 0, downgrad 1, weak 0	**5 tokens**
Sentence C	revenue 1, downgrad 1, weak 0	**5 tokens**
Sentence D	revenue 1, downgrad 1, weak 2	**6 tokens**
Column totals	revenue 3, downgrad 3, weak 2	**21 tokens**

IN THE LESSON

- Example two.
- Dana has four sentences, already normalised.
- Sentence A: revenue, rise, margin, stay, strong.
- Sentence B: not, expect, downgrad, leverag, fall.
- Sentence C: revenue, fall, downgrad, seem, close.
- Sentence D: downgrad, follow, weak, revenue, weak, margin.
- We show just three columns: revenue, downgrad and weak.
- Sentence A has revenue once and the other two zero.
- Sentence B has downgrad once.
- Sentence C has revenue once and downgrad once.
- Sentence D has revenue once, downgrad once and weak twice, because weak appears twice, and it has six tokens in all while the others have five.
- The bottom row adds each column: revenue three, downgrad three, weak two.
- Across all four sentences there are twenty one tokens.

7 N-grams: putting some order back

- Unigram 1 token, bigram 2, trigram 3
- Bigram keeps phrases such as 'not profitable'
- 5 tokens make 4 bigrams (n − 1)
- Cost: many more columns

IN THE LESSON

- A bag of words forgets order, and order sometimes matters.
- Compare profitable with not profitable.
- A bag of single words sees profitable and not as separate tokens, so it cannot tell that the second phrase is a warning.
- The fix is an n gram, a run of n tokens kept together.
- A unigram is one token.
- A bigram is two in a row.
- A trigram is three.
- Take Sentence B, with five tokens: not, expect, downgrad, leverag, fall.
- It makes four bigrams: not expect, expect downgrad, downgrad leverag, and leverag fall.
- In general, five tokens give five minus one bigrams.
- The price is size, because every new n gram is another column and the matrix grows quickly.

8 Which tokens deserve to stay?

- TF: count in a sentence ÷ tokens in it
- DF: share of sentences containing the token
- TF-IDF: TF times a boost for rarity
- High = common here, rare elsewhere

IN THE LESSON

- With thousands of columns, which tokens deserve to stay as features?
- Imagine a bookshop.
- If every book contains the word the, finding that word tells you nothing about which book you hold.
- But if a word such as impairment appears in only one book, finding it says a lot.
- Three measures capture this.
- Term frequency, T F, is how often a token appears in one sentence, as a share of that sentence's tokens.
- Document frequency, D F, is the share of all sentences that contain the token at all.
- And T F I D F multiplies the first by a boost for rarity.
- A token scores high when it is common in this sentence but rare in the others.

9 TF-IDF, term by term

TF-IDF = TF × ln(1 ÷ DF)

TF	Count in sentence ÷ its tokens
DF	Share of sentences with it
ln(1 ÷ DF)	Rarity boost, big if DF small
TF-IDF	Frequent here, rare elsewhere

IN THE LESSON

- Here are the pieces.
- T F is the count of the token in the sentence, divided by the number of tokens in that sentence.
- D F is the number of sentences containing the token, divided by the total number of sentences.
- The rarity boost, called inverse document frequency, is the natural log of one divided by D F.
- Look at what it does.
- If a token is in every sentence, D F is one, and the natural log of one is zero, so the score is zero whatever T F is.
- If a token is in one sentence of four, D F is one quarter, one divided by one quarter is four, and the natural log of four is about one point three eight six.
- Rarer tokens get a bigger boost.
- Multiply by T F and you have the score.

10 Example 3: TF-IDF of weak

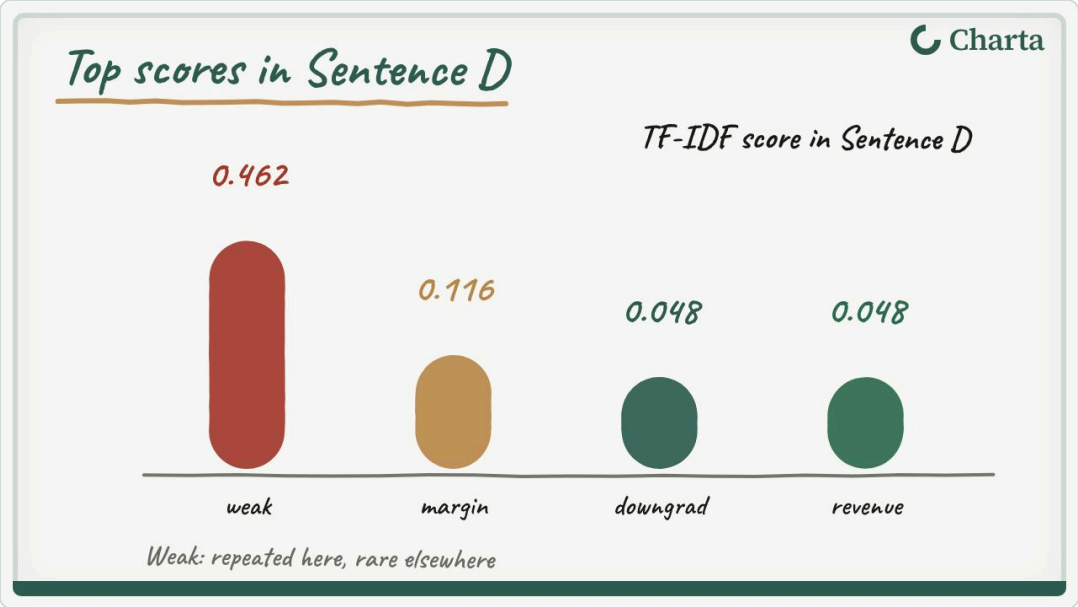
Sentence D: downgrad follow weak revenue weak margin

TF of weak	2 ÷ 6	**0.333**
DF of weak	1 sentence in 4	**0.25**
Boost ln(1 ÷ 0.25)	ln 4	**1.386**
TF-IDF of weak	0.333 × 1.386	**0.462**
TF-IDF of downgrad	0.167 × 0.288	**0.048**

IN THE LESSON

- Example three.
- Stay with Sentence D, which has six tokens, and score the token weak.
- Step one, term frequency.
- Weak appears twice among six tokens, so T F is two divided by six, about zero point three three three.
- Step two, document frequency.
- Weak appears in one sentence out of four, so D F is zero point two five.
- Step three, the boost: one divided by zero point two five is four, and the natural log of four is one point three eight six.
- Step four, multiply: zero point three three three times one point three eight six gives about zero point four six two.
- For comparison, downgrad is in three sentences of four, so its boost is only zero point two eight eight, and its score is about zero point zero four eight.

11 Top scores in Sentence D



IN THE LESSON

- Here are four tokens from Sentence D drawn as bars.
- Up the side is the T F I D F score, and along the bottom are the tokens.
- The tall bar is weak, at zero point four six two.
- It is repeated twice in this sentence and it appears nowhere else.
- Next is margin, at zero point one one six.
- It appears in two of the four sentences, so its boost is the natural log of two, and one sixth of that is the score.
- The last two are downgrad and revenue, both at about zero point zero four eight.
- They each appear in three of the four sentences, so their boost is small.

12 Feature engineering: new features

- Selection: keep the useful tokens
- Engineering: build new features from the text
- Numbers, named entities, parts of speech
- Each new feature is one more column

IN THE LESSON

- Choosing which tokens to keep is feature selection.
- Feature engineering is different: it means creating new features from the text, rather than only keeping the tokens you started with.
- Dana has three useful ideas.
- First, numbers.
- A figure in the text, such as a percentage, can be pulled out and kept as an actual number, because debt levels and growth rates matter a great deal to credit.
- Second, named entities.
- Tag whether a word names a company, a person, a place or a date, so the model can tell Kestrel the firm from kestrel the bird.
- Third, parts of speech.
- Tag each token as a noun, a verb or an adjective, so the model can treat downgrade the verb differently from downgrade the noun.
- Each new feature is one more column.

13 Cleansing: wash before you cook

- Remove HTML tags
- Remove most punctuation and extra white space
- Replace every number with a /number/ token
- Then tokenise and normalise

IN THE LESSON

- Now let us go back to the start of the pipeline and slow down, because most of the quality in a text project is decided before any model sees a word.
- Think of washing vegetables before you cook them.
- Dana's transcripts come off a web page, so they arrive with leftovers.
- Cleansing removes those.
- First, the formatting code: H T M L tags, the little instructions in angle brackets that tell a browser how to show a page.
- Second, most punctuation, because full stops and commas rarely help a counting model.
- Third, extra white space.
- Fourth, numbers: replace each one with a single placeholder token, written slash number slash.
- That way three, three hundred and three million all count as the same kind of thing.
- Only after that do we tokenise and normalise.

14 Example 4: cleansing a scraped line

* Not a downgrade after 3 strong quarters. *

Strip the HTML tags	drop and	**7 words**
Remove punctuation	drop the full stop	**7 words**
Replace the number	3 becomes /number/	**7 tokens**
Lower case	Not becomes not	**7 tokens**

IN THE LESSON

- Example four.
- Dana's scraper brings back this line, still wrapped in web page tags: not a downgrade after three strong quarters, with a capital N on not.
- Step one, strip the tags.
- We drop the opening and closing paragraph tags and are left with seven words.
- Step two, remove punctuation.
- The full stop goes, and we still have seven.
- Step three, replace the number three with the placeholder, so the line now carries the placeholder as one token.
- Still seven tokens, because the placeholder simply takes the place of the three.
- Step four, lower case, so the capital N becomes a small n.
- Notice that cleansing removed junk without removing a single real word.
- The next stage, normalisation, is where words begin to disappear.

15 Example 5: the stop list that flips meaning

not a downgrade after /number/ strong quarters

Cleaned line	not, a, after are stop words	**7 tokens**
Standard stop list	drops not, a, after	**4 tokens**
What the model sees	downgrad /number/ strong quarter	**looks bad**
Custom list keeping not	not stays in the sentence	**5 tokens**

IN THE LESSON

- Example five, and it is a trap the examiners love.
- A stop word list is just a list somebody wrote, and many standard lists include not, no and never, because they are tiny and common.
- Take Dana's cleaned line: not, a, downgrade, after, number, strong, quarters.
- That is seven tokens.
- Run a standard stop list, which removes not, a and after, and stem what is left.
- Four tokens remain: downgrad, number, strong, quarter.
- Read that back and it looks like bad news.
- But the original said there was not a downgrade.
- The negation was thrown away and the meaning reversed.
- Now use a custom list that keeps not.
- Five tokens remain, and the sentence still carries its message.
- Choosing the stop list is a modelling decision, not a chore.

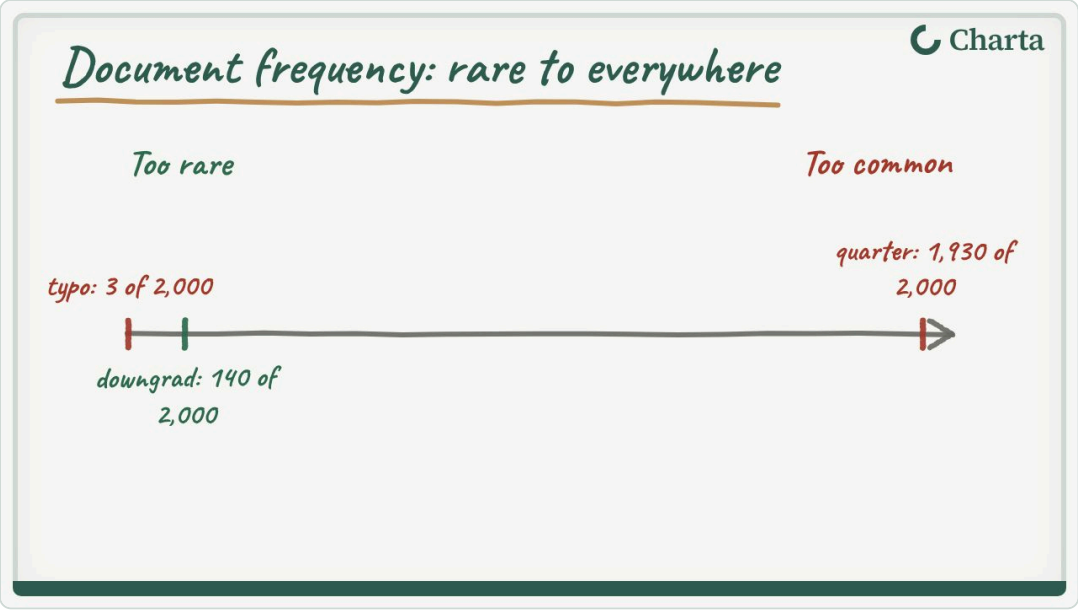
16 Exploring: count before you model

- Count tokens, and the sentences containing them
- Word cloud: bigger word, more frequent
- Too common: no power to tell sentences apart
- Too rare: sparse noise. Remove both

IN THE LESSON

- Once the text is clean and tokenised, Dana explores it before modelling, the way a cook tastes the soup before serving it.
- The simplest exploration is counting: how often does each token appear in total, and in how many sentences?
- A quick way to see the counts is a word cloud, a picture in which the more frequent a token, the bigger it is drawn.
- It is fast, but it shows only frequency.
- Counting reveals two problems.
- Tokens that appear in nearly every sentence, such as quarter or company, carry almost no distinguishing information.
- Tokens that appear only once or twice are so rare that they add sparse noise.
- Both groups are usually removed.

17 Document frequency: rare to everywhere



IN THE LESSON

- Here are three tokens placed on a line by their document frequency, the share of sentences that contain them.
- The line runs from too rare on the left to too common on the right.
- On the far left is a typing error that appears in only three of the two thousand sentences, a document frequency of zero point zero zero one five.
- Three examples teach a model nothing dependable, so it gets pruned.
- On the far right is quarter, in one thousand nine hundred and thirty of the two thousand sentences, a document frequency of zero point nine six five.
- It is everywhere, so it cannot separate the classes, with a T F I D F boost of only about zero point zero three six.
- In between sits downgrad, in one hundred and forty sentences, a document frequency of zero point zero seven.
- That is the useful middle.
- The cut offs are the analyst's choice.

18 Ranking what is left

- Chi-square: are token and class independent?
- Mutual information: how much does the token tell?
- Zero if spread evenly across classes
- Large if concentrated in one class: rank high

IN THE LESSON

- After pruning, many tokens remain.
- How do we rank them by usefulness for telling downgrades from the rest?
- There are two standard tools.
- A chi square test checks whether a token's presence and the class label are independent.
- Build a small table counting sentences with and without the token, split by class.
- Compare the observed counts with the counts you would expect if the token and the class were unrelated.
- A big gap gives a big chi square and a high rank.
- Mutual information asks the same question in different words: how much does knowing the token is present tell you about the class?
- It is zero when the token is spread evenly across the classes and grows as the token concentrates in one.
- Rank by either, and keep the top tokens.

19 The chi-square statistic

Chi-square = Σ (O − E)² ÷ E

O	Count actually seen in a cell
E	Count if token and class unrelated
Σ	Add over all four cells
Table value	Compare: 3.84 for one df, 5%

IN THE LESSON

- For each of the four cells in the table, take the observed count O, subtract the expected count E, square the gap, and divide by E.
- Add the four results.
- The expected count for a cell is its row total times its column total, divided by the grand total.
- Squaring makes every gap positive.
- A large total means the token and the class are probably not independent.
- To decide how large is large, compare with a critical value from a table.
- For a two by two table, which has one degree of freedom, at the five per cent level, the critical value is three point eight four.
- Above that, we reject independence, and the token is informative.

20 Example 6: ranking downgrad

2,000 sentences: 1,000 negative, 1,000 positive (made up)

Observed, token present	128 negative, 12 positive	**140**
Expected if unrelated	70 negative, 70 positive	**140**
Present cells, each	58² ÷ 70	**48.06 + 48.06**
Absent cells, each	58² ÷ 930	**3.62 + 3.62**
Chi-square total	far above 3.84: keep it	**103.35**

IN THE LESSON

- Example six, with made up counts.
- Of Dana's two thousand sentences, one thousand are labelled negative and one thousand positive.
- The token downgrad is present in one hundred and forty: one hundred and twenty eight negative and twelve positive.
- It is absent from the other one thousand eight hundred and sixty, eight hundred and seventy two negative and nine hundred and eighty eight positive.
- If the token were unrelated to the class, the present row would split seventy and seventy, and the absent row nine hundred and thirty each.
- Every observed count is fifty eight from its expected count.
- Each present cell gives fifty eight squared over seventy, about forty eight point oh six.
- Each absent cell gives fifty eight squared over nine hundred and thirty, about three point six two.
- The total is about one hundred and three point three five, far above three point eight four, so downgrad ranks high.
- For quarter, every gap is zero.

21 Four mistakes to avoid

- 1. Rows are documents, columns are tokens
- 2. High TF-IDF: common here, rare elsewhere
- 3. A stop list can delete 'not' and flip meaning
- 4. Prune the too common and the too rare

IN THE LESSON

- Before we turn to training, here are four slips that cost marks.
- One, swapping the rows and columns of the matrix.
- Rows are documents, columns are tokens.
- Two, reading T F I D F backwards.
- A high score means frequent in this sentence and rare across the others, not simply frequent.
- A token in every sentence scores zero.
- Three, letting a stop list delete not.
- Normalisation must never destroy meaning, especially before sentiment analysis.
- Four, keeping the extremes.
- Tokens in almost every sentence carry no distinguishing information, and very rare tokens only add noise, so both are usually removed before ranking with chi square or mutual information.

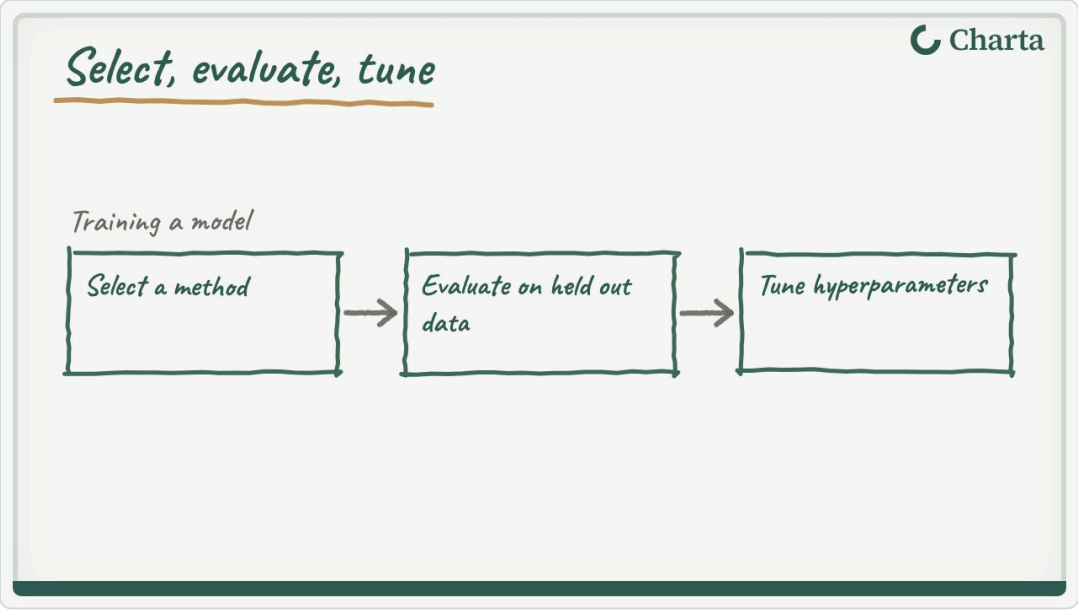
22 Training a model: three stages

- 1. Method selection: a method that fits the problem
- 2. Evaluation: judge on data it has not seen
- 3. Tuning: adjust hyperparameters, repeat
- Hyperparameter: a setting you choose, not learn

IN THE LESSON

- Dana's table of numbers is ready.
- Now she must train a model, which means letting a method learn the link between the features and the labels.
- Think of teaching a chef a new dish.
- First, you choose a style of recipe that suits the ingredients.
- Second, you serve it to a guest who has not tasted it before and judge honestly.
- Third, you adjust the settings, such as oven heat and cooking time, and try again.
- Those are the three stages of model training: method selection, performance evaluation and tuning.
- A setting you choose yourself, rather than one the model learns from the data, is called a hyperparameter.
- The strength of a penalty and the depth of a decision tree are two examples.

23 Select, evaluate, tune



IN THE LESSON

- Here are the three stages as a loop.
- On the left, select a method.
- That depends on facts about the problem, which we meet next.
- In the middle, evaluate it.
- Measure performance with the right metrics, on data kept back from training, using cross validation to make the most of a limited sample.
- On the right, tune.
- Adjust the hyperparameters, often by grid search, which means trying a range of settings laid out in a grid and keeping the best.
- Then go round again.
- The board shows the arrows pointing forward, but in practice the validation results send you back to change the settings, until the validation error stops improving.

24 Choosing the method

Facts about the problem decide the method

Situation	Method family
Labelled, target is a number	Regression
Labelled, target is a category	Classification
No labels, find groups	Clustering
Huge, messy text or images	Neural networks

IN THE LESSON

- Four facts decide the method.
- First, are the data labelled?
- If yes, this is supervised learning.
- If no, it is unsupervised, and we look for groups or reduce the number of features.
- Second, what type is the target?
- A number, such as next year's yield, points to regression.
- A category, such as downgrade or no downgrade, points to classification.
- Dana's data are labelled with a category, so she needs supervised classification.
- Third, the type and number of features, since hundreds of thousands of token columns may need reducing first.
- Fourth, the size of the dataset.
- Very large, messy data may justify a neural network, while a small sample may not support one.

25 Example 7: accuracy can mislead

Test: 500 transcripts, 30 downgrades. Model: 21 caught, 24 false alarms

Always say no downgrade	470 right of 500	**94.0%**
Its recall	0 caught of 30	**0%**
Our model: accuracy	467 right of 500	**93.4%**
Our model: recall	21 caught of 30	**70%**
Our model: precision	21 right of 45 alerts	**46.7%**

IN THE LESSON

- Example seven.
- Dana holds back five hundred transcripts for testing, and thirty of them were followed by a downgrade.
- Consider a lazy model that always says no downgrade.
- It is right on four hundred and seventy, so its accuracy is ninety four per cent.
- That sounds great, yet it catches none of the thirty downgrades.
- Now her real model.
- It catches twenty one downgrades, misses nine, raises twenty four false alarms, and correctly clears four hundred and forty six.
- Right answers number four hundred and sixty seven, an accuracy of ninety three point four per cent, lower than the lazy model.
- But recall, the share of real downgrades caught, is twenty one over thirty, seventy per cent.
- Precision, the share of alerts that were real, is twenty one over forty five, about forty six point seven per cent.
- Choose metrics that notice the rare class.

26 Example 8: cross-validation

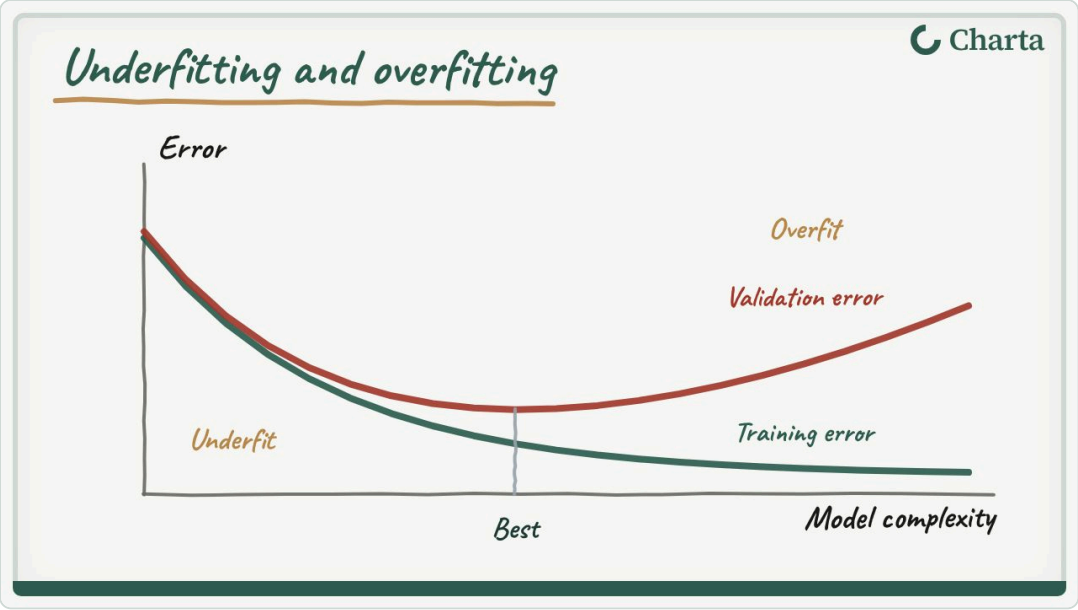
1,500 training transcripts split into 5 folds of 300

Fold 1 held out	train on 1,200	**12%**
Fold 2 held out	train on 1,200	**9%**
Fold 3 held out	train on 1,200	**11%**
Folds 4 and 5 held out	train on 1,200 each time	**8% and 10%**
Average validation error	(12+9+11+8+10) ÷ 5	**10%**

IN THE LESSON

- Example eight.
- Dana does not touch the test set while she trains.
- She uses the other one thousand five hundred transcripts, ninety of which were followed by a downgrade.
- To judge a model without wasting data, she uses five fold cross validation.
- She splits the one thousand five hundred into five folds of three hundred.
- In turn she trains on four folds, which is twelve hundred transcripts, and measures the error on the fold held out.
- Every fold gets one turn as the validation sample.
- Her five errors are twelve, nine, eleven, eight and ten per cent.
- The average is ten per cent.
- That average is a steadier estimate than any single split, because every transcript has been used for training and for validating, though never in the same round.

27 Underfitting and overfitting



IN THE LESSON

- Now tuning, which is a balancing act between two failures.
- Here is the picture.
- Along the bottom is model complexity, how flexible the model is, and up the side is error.
- Start with the training error, the falling line.
- A more flexible model always fits its training data better, so this line keeps dropping.
- Now the validation error, the second line.
- At first it falls too, because the model is learning real patterns.
- Then it turns upward, because the model starts memorising noise that will not repeat.
- On the left, both errors are high and close together.
- That is underfitting: the model is too simple.
- On the right, training error is low but validation error is high and the gap is wide.
- That is overfitting.
- The best setting is the lowest point of the validation line.

28 Example 9: a grid search

Penalty setting: training error and validation error

Penalty 0.01	train 3%, validation 15%	**gap 12: overfit**
Penalty 0.1	train 8%, validation 10%	**gap 2**
Penalty 1	train 18%, validation 19%	**underfit**
Chosen: penalty 0.1	lowest validation error	**10%**

IN THE LESSON

- Example nine.
- Dana's classifier has a penalty hyperparameter that punishes complexity.
- A bigger penalty gives a simpler model.
- She tries three settings in a grid.
- With a penalty of zero point zero one, training error is three per cent but validation error is fifteen, a gap of twelve points.
- That is overfitting.
- With a penalty of zero point one, training error is eight and validation error is ten, a gap of just two.
- With a penalty of one, training error is eighteen and validation error is nineteen.
- Both are high and close together, which is underfitting.
- The rule is to choose the setting with the lowest validation error, here zero point one, at ten per cent.
- Never choose by training error, which would pick the overfitted model.

29 Example 10: unbalanced classes

Training set: 1,500 transcripts, 90 are downgrades (6%)

Starting point	90 downgrades, 1,410 not	**6% minority**
Oversample the minority	90 copied up to 1,410	**2,820 rows**
Undersample the majority	keep 90 of the 1,410	**180 rows**
Test set	left alone: 30 of 500	**6% minority**

IN THE LESSON

- Example ten.
- In the one thousand five hundred training transcripts, only ninety are followed by a downgrade, six per cent.
- A model trained on this can look good by ignoring the rare class.
- There are two common fixes.
- Oversampling the minority copies the ninety downgrade cases until there are as many as the one thousand four hundred and ten others, which gives two thousand eight hundred and twenty rows.
- Undersampling the majority keeps only ninety of the one thousand four hundred and ten, giving one hundred and eighty rows, but it throws data away.
- Now the crucial rule: do this to the training data only.
- The test set stays as it is, with its natural six per cent of downgrades, so it reflects the real world.
- Rebalance the test set and the scores will flatter the model.

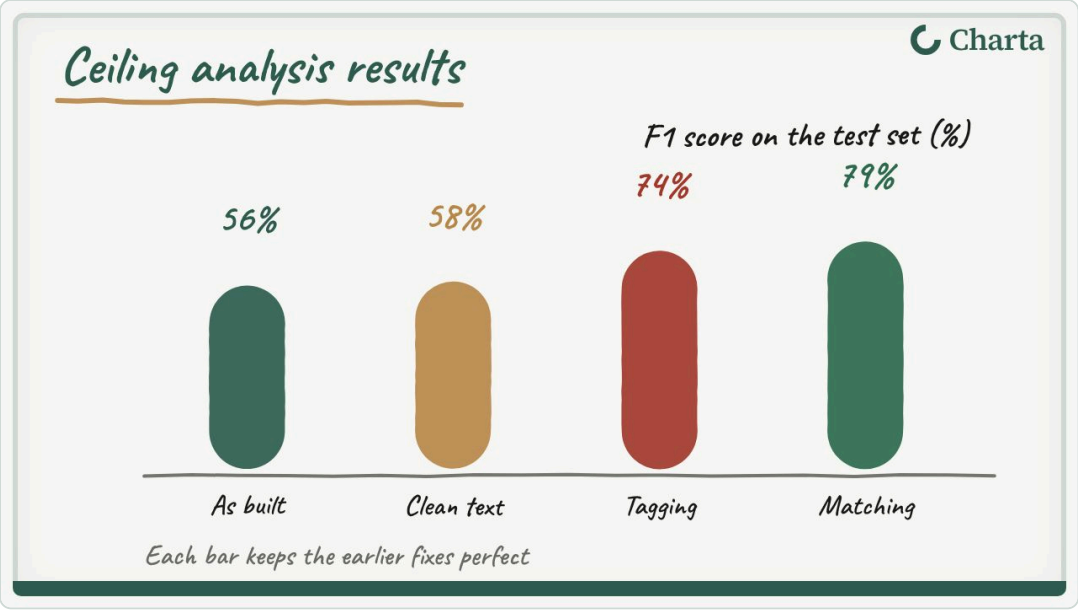
30 Ceiling analysis: where to spend effort

- Pipeline: several parts, one feeding the next
- Make each part perfect in turn, and keep it perfect
- Measure how much the score rises each time
- Spend effort where the rise is biggest

IN THE LESSON

- When a model is still not good enough, where should the team spend its effort?
- Many real systems are pipelines of several parts, each doing one job, with the output of one feeding the next.
- Ceiling analysis finds the part that limits performance most.
- For each part, in pipeline order, you replace it with perfect output, often produced by hand, keep it perfect, and measure how much the overall score rises.
- The part whose fix lifts the score most is where improvement is worth chasing.
- Dana's pipeline has four parts: text cleaning, a tagging module, a company matching module, and the classifier.
- As built, the F one score on the test is fifty six per cent.
- That is the single score that combines precision and recall from Example seven.

31 Ceiling analysis results



IN THE LESSON

- Here is the result as bars, with the F one score up the side.
- As built, fifty six per cent.
- Now make text cleaning perfect and the score rises to fifty eight, a gain of just two points.
- Keep that, and make the tagging module perfect as well, and it jumps to seventy four, a gain of sixteen.
- Keep both, and make company matching perfect, and it reaches seventy nine, a further five.
- The tagging module is the bottleneck.
- Dana should spend her next weeks there, and not on cleaning, which could add at most two points.
- Notice that each bar includes the fixes before it, so the gains are cumulative.
- We stop before the classifier itself, because a perfect classifier would trivially score one hundred.

32 Recap

- Rows = documents, columns = tokens
- Cleanse, normalise (keep 'not'), prune, rank
- Select, evaluate on held out data, tune
- Resample training data only; ceiling analysis

IN THE LESSON

- Let us pull it together.
- Text becomes features in stages: tokenise, normalise, count in a bag of words, and stack the counts in a document term matrix with documents as rows and tokens as columns.
- N grams keep short phrases.
- T F I D F is term frequency times the natural log of one over document frequency.
- Engineered features add numbers, entities and parts of speech.
- Preparing text means cleansing first, then normalising without destroying meaning, then exploring, pruning the too common and the too rare, and ranking by chi square or mutual information.
- Training has three stages: select a method, evaluate on held out data with cross validation, and tune hyperparameters.
- Fix unbalanced classes in the training data only, and use ceiling analysis to find the weak part of a pipeline.

33 Try three quick questions

1. A token appears 3 times in a 12-token sentence and in 1 of 5 sentences. TF-IDF?
   Answer: 0.25 × ln 5 = 0.25 × 1.609 = 0.402
2. Why can a standard stop list damage a sentiment model?
   Answer: It can delete 'not', so the meaning is flipped.
3. Training error 18%, validation error 19%. What is wrong, and what next?
   Answer: Underfitting: a more flexible model, e.g. a smaller penalty.

IN THE LESSON

- Now try three quick questions yourself.
- Pause after each one.
- Question one.
- A token appears three times in a twelve token sentence, and in one of five sentences.
- What is its T F I D F?
- The T F is three over twelve, a quarter.
- The boost is the natural log of five, about one point six oh nine.
- Multiply: about zero point four oh two.
- Question two.
- Why can a standard stop list damage a sentiment model?
- It can delete the word not, so bad news reads as the opposite.
- Question three.
- Training error is eighteen per cent and validation error is nineteen per cent.
- What is wrong, and what next?
- Both are high and close together, so the model is underfitting.
- Make it more flexible, for example with a smaller penalty.