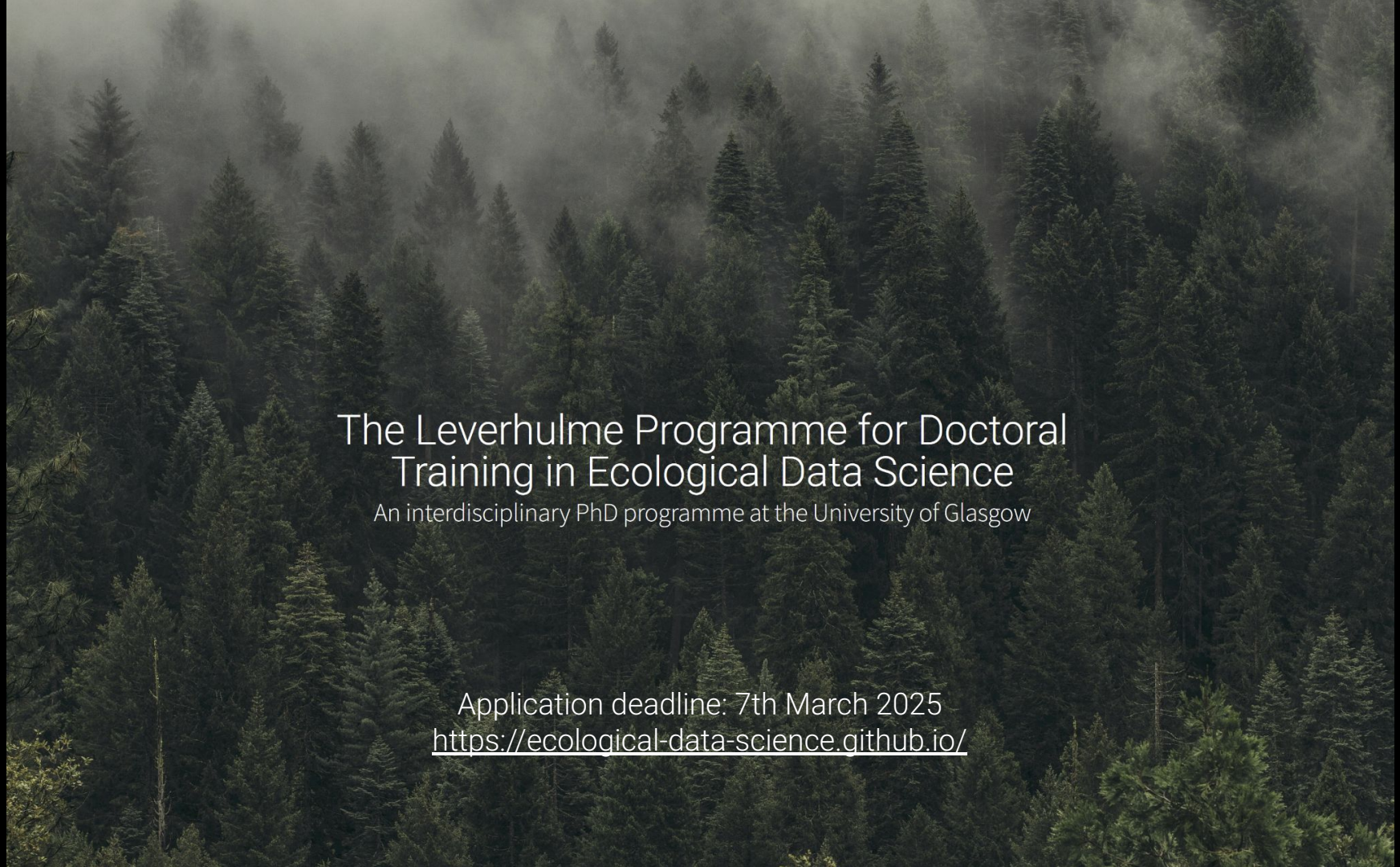
PhDs at Glasgow

- Join an international community of researchers
- Flexible working with a friendly work environment and fantastic compute resources
- Pursue your curiosity and become a world-expert on an NLP problem
- PhD students are funded with a modest stipend
- No need for an MSc/MSci

Come chat with me if interested in a PhD


The Leverhulme Programme for Doctoral Training in Ecological Data Science
An interdisciplinary PhD programme at the University of Glasgow
Application deadline: 7th March 2025
https://ecological-data-science.github.io/

Text Classification

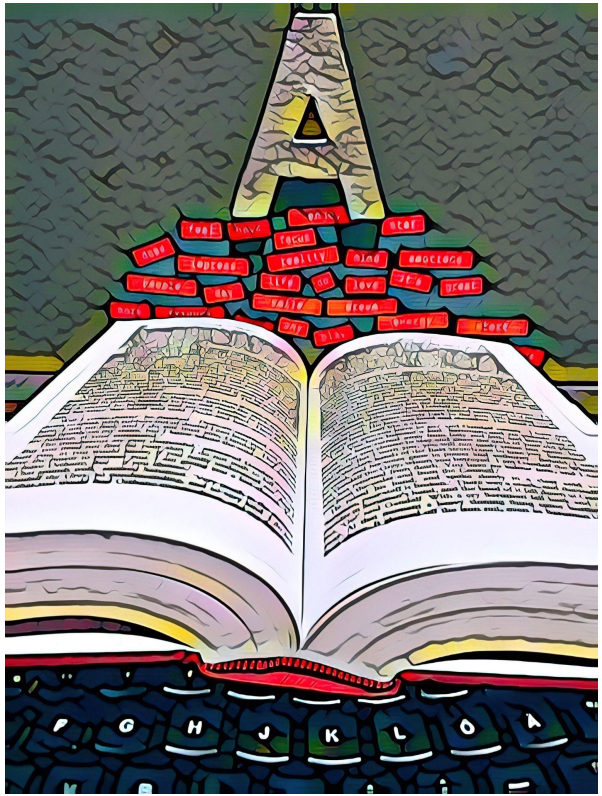
Sean MacAvaney & Jake Lever

University of Glasgow | School of Computing Science

Today's Main Topics

- Machine Learning for Text Classification
  - "Which class of thing does this text belong to?"

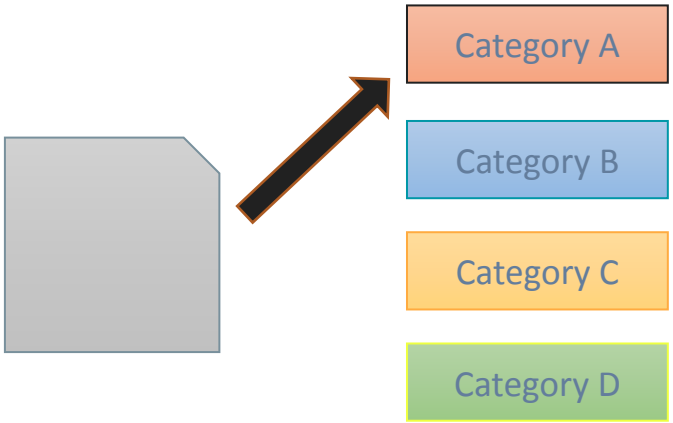
- Classification Evaluation
  - "How good are these class predictions?"



https://betterimagesofai.org

Classification

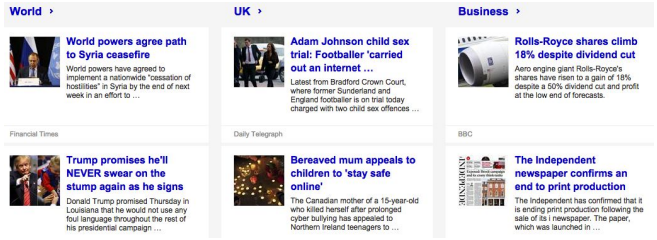
- The task of **predicting** which of a predefined **set of classes** (or **categories**) an object belongs to:
- In other words, assigning a **class** to an **object** based on some **pre-trained** knowledge:

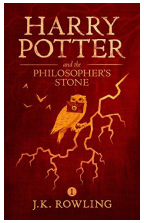


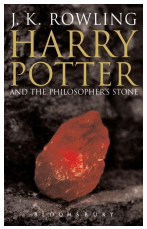
Our objects to classify

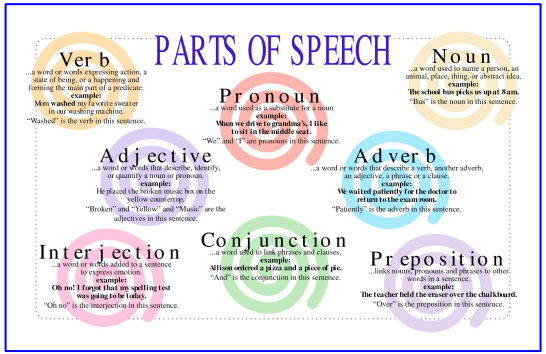
In **our context** we would work with:

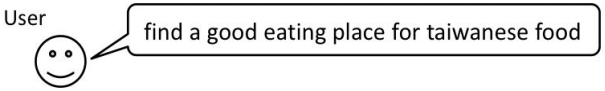
- Web pages
- Text documents (Such as, News articles)
- Emails
- Product descriptions
- Natural Language questions
- Social Media Posts
- Words themselves
- **Really anything with words…**











We'll be using Sklearn in Python: https://scikit-learn.org/stable/user_guide.html

Example: What is the subject of this article?

MeSH Subject Category Hierarchy

- Antagonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- …

?

Subject classification can help users identify which articles are potentially interesting to them.

MEDLINE Article

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Brain and Cognition 53 (2003) 223–228
Brain and Cognition
www.elsevier.com/locate/b&c

Syntactic frame and verb bias in aphasia: Plausibility judgments of undergoer-subject sentences

Susanne Gahl,[a] Lise Menn,[b] Gail Ramsberger,[b] Daniel S. Jurafsky,[b] Elizabeth Elder,[b] Molly Rewega,[c] and L. Holland Audrey[c]

[a] *Harvard University, Cambridge MA, USA*
[b] *University of Colorado, Boulder CO, USA*
[c] *University of Arizona, Tucson AZ, USA*

Accepted 7 May 2003

Abstract

This study investigates three factors that have been argued to define "canonical form" in sentence comprehension: Syntactic structure, semantic role, and frequency of usage. We first examine the claim that sentences containing unaccusative verbs present difficulties analogous to those of passive sentences. Using a plausibility judgment task, we show that a mixed group of aphasics performed significantly better on unaccusatives than on passives. We then turn to the observation that passives are generally harder than actives for aphasics. We show that this effect is modulated by lexical bias, i.e., the likelihood that a verb appears in a given syntactic structure: Passives of passive-bias verbs were significantly easier than passives of active-bias verbs. More generally, sentences whose structure matches the lexical bias of the main verb are significantly easier than sentences in which structure and lexical bias do not match. These findings suggest that "canonical form" reflects frequency and lexical biases.

1. Introduction

The simplicity of "canonical form," or "canonical word order," for normal and aphasic comprehension has often been taken as self-evident in the sentence comprehension literature. However, as has been pointed out by Menn (2000), the privileged status of canonical form itself needs explanation. Different definitions of "canonical form" yield testably different predictions. One approach to the definition of canonical sentence form is that implicit in Bates, Friederici, and Wulfeck (1987, *inter alii*). Bates et al. note that sentences with Agent–Action–Object order represent the canonical word order for English. A second approach is based on syntactic "movement" analyses and defines as non-canonical any word order that diverges from the [$_{IP}$NP-[$_{VP}$Verb-NP]] configuration assumed for the deep structure of English sentences. Based on this understanding of canonicity, Kegl (1995) argues that sentences with unaccusative verbs should be difficult to process for aphasic patients, in particular for patients with "agrammatism," for reasons that are analogous to the factors giving rise to the greater difficulty of passives compared to actives. Although the precise definition of unaccusativity is contested (see e.g., Levin & Rappaport Hovav, 1995), unaccusative verbs are generally understood to be intransitive verbs whose (surface) subjects represent Undergoer arguments. Examples of unaccusative verbs include verbs like *melt* and *blush*. Under the transformational analysis assumed in Kegl (1995), the surface subjects of unaccusative verbs are linked via movement to direct objects in deep structure. Unaccusatives therefore induce the very same difficulties as passive sentences, according to Kegl's analysis, and should be as hard as passives for aphasic speakers.

A different approach to canonical form has been proposed by Menn et al. (1998) who suggest that canonical form relies on the most frequent syntactic frame *for a given verb*. Under this view, aphasic problems with producing and understanding passives derive from the fact that, for most transitive verbs, passives occur less frequently than actives. One prediction of this approach, also advanced by Gahl (2002), is that comprehension difficulty should vary with the lexical bias of the words

0278-2626/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0278-2626(03)00114-3

https://www.nlm.nih.gov/mesh/meshhome.html

Example: YouTube comment spam detection

?

CONTENT	CLASS
Huh, anyway check out this you[tube] channel: kobyoshi02	1
Hey guys check out my new channel and our first vid THIS IS US THE MONKEYS!!! I'm the monkey in the white shirt,please leave a like comment and please subscribe!!!!	1
just for test I have to say murdev.com	1
me shaking my sexy ass on my channel enjoy ^_^	1
watch?v=vtaRGgvGtWQ Check this out .	1
Hey, check out my new website!! This site is about kids stuff. kidsmediausa . com	1
Subscribe to my channel	1
i turned it on mute as soon is i came on i just wanted to check the views...	0
You should check my channel for Funny VIDEOS!!	1
and u should.d check my channel and tell me what I should do next!	1

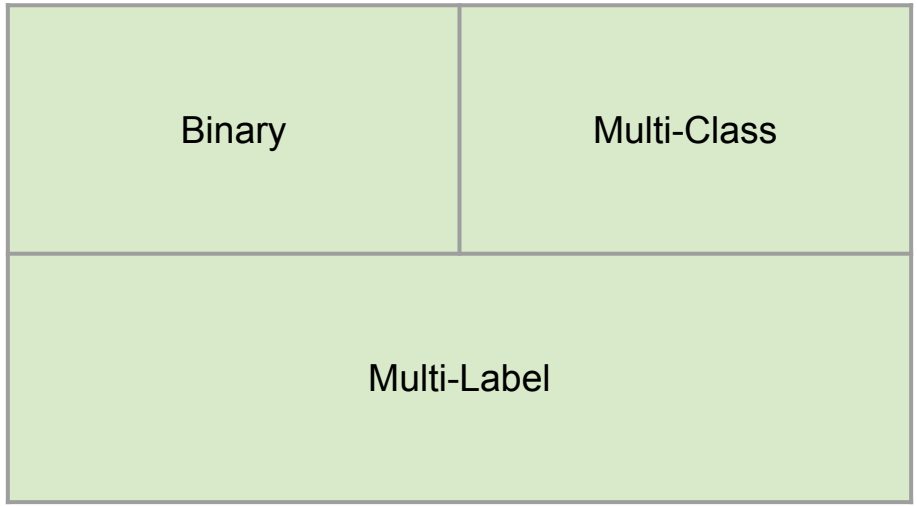
Spam classification can help filter out comments that users do not want to see.

Applications of Text Classification

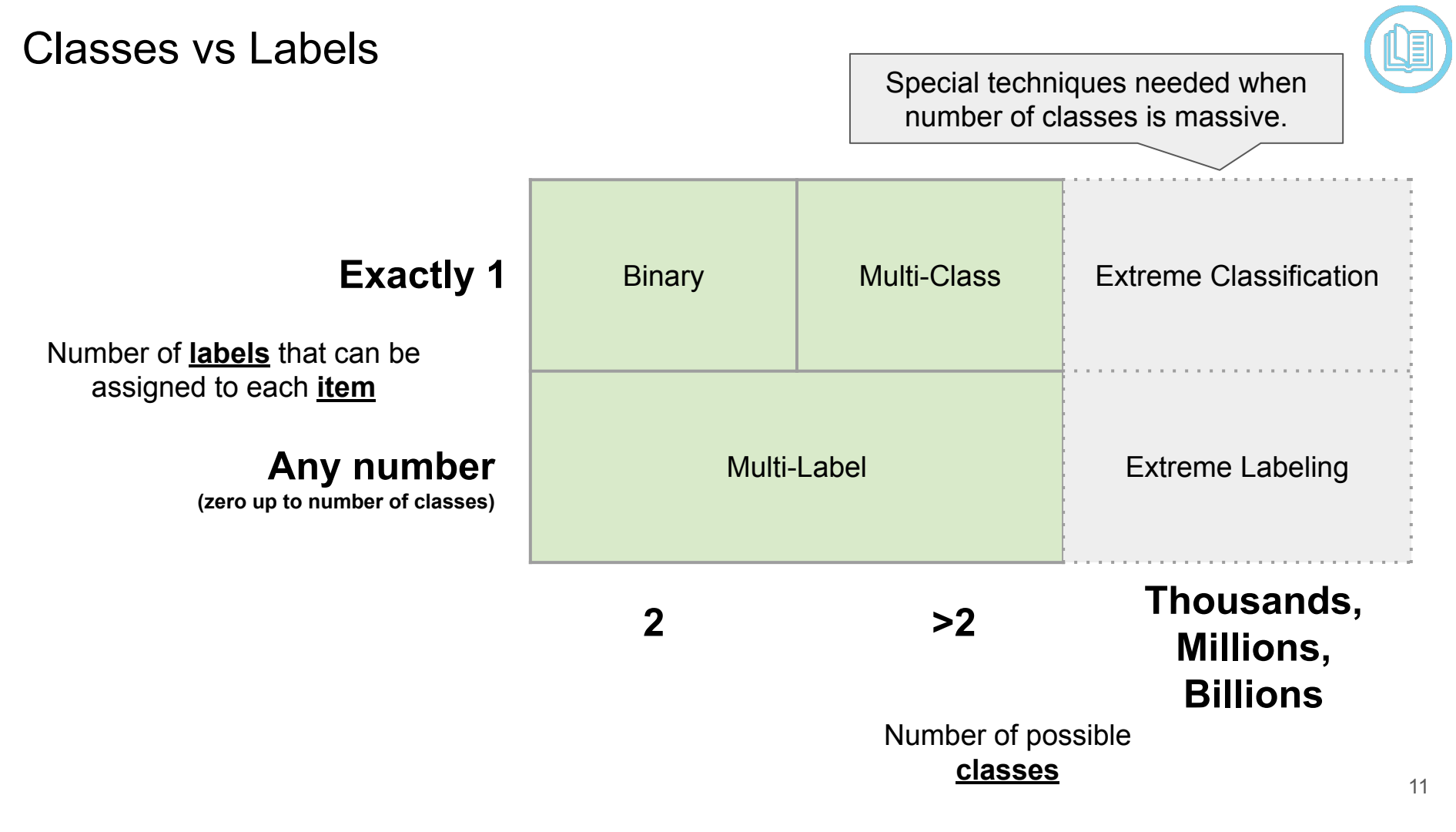
Text Classification is an essential element in a wide range of modern tasks:

- **Topic categorisation**: Is this document about [technology / sports / lifestyle, etc..]?
- **Spam detection**: Is this comment/email spam?
- **Sentiment analysis**: Does this tweet express a negative opinion?
- **Language identification**: Is this document written in Spanish?
- **Author attribution**: Is this document written by John Smith?
- **Domain-specific**: Is this content “Clinical Trial Results” vs “Not Clinical Trial Results”?
- **Discourse:** Is this Reddit post a question, an answer or clickbait?
- and many more!

Classes vs Labels



Classes vs Labels



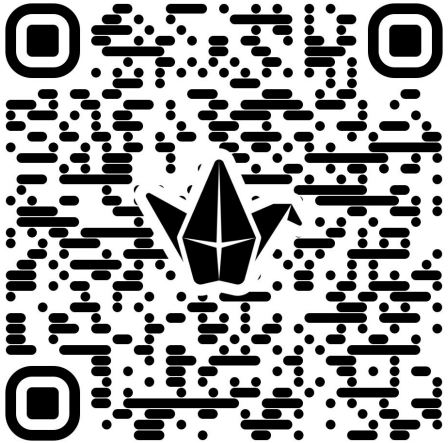
Think-Pair-Share: Binary, multiclass and multilabel problems

Discuss in your group and come up with text classification problems that are:

- Binary (predicting one of two possible classes)
- Multi-class (predicting one of three or more possible classes)
- Multi-label (predicting zero, one, two or more classes)

Example text: news articles, social media posts, restaurant reviews, etc

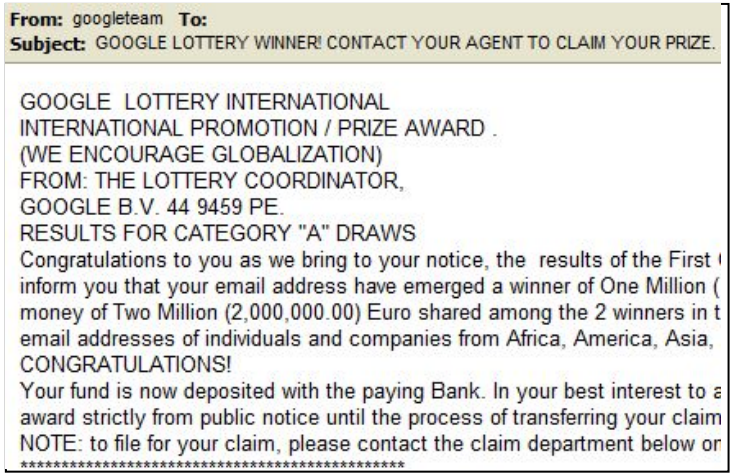
https://padlet.com/jakelever/tad2025_5

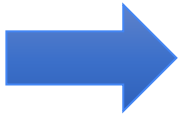


Binary classification

Each item belongs to exactly 1 out of 2 classes

Spam vs Not Spam





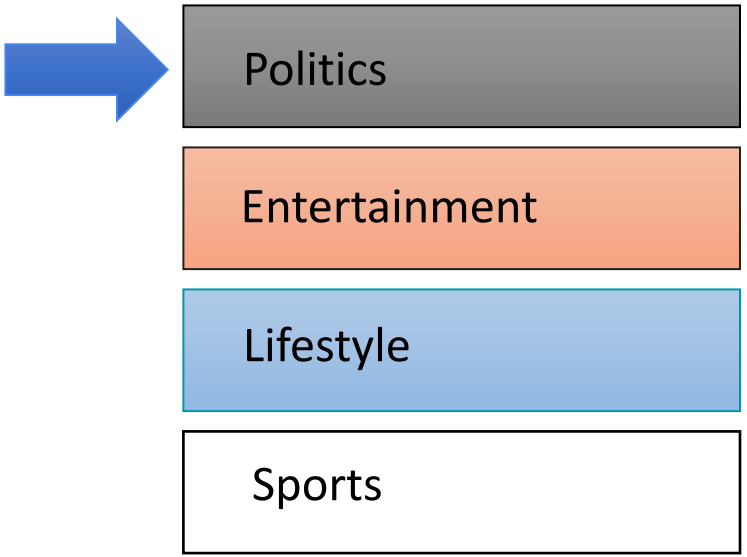
Spam

Not Spam

Multi-Class

Each item belongs to exactly 1 out of >2 classes

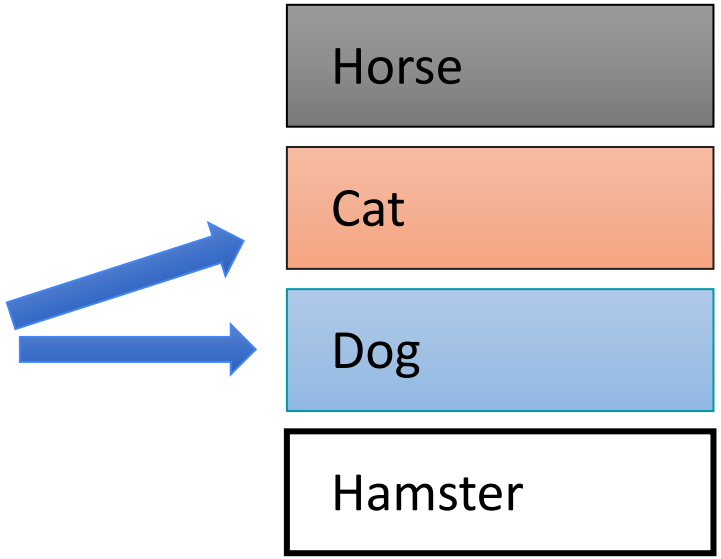
Categorisation



Multi-Label

Each item can belong to 0, 1 or Many classes

Topic Labeling



https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0237822

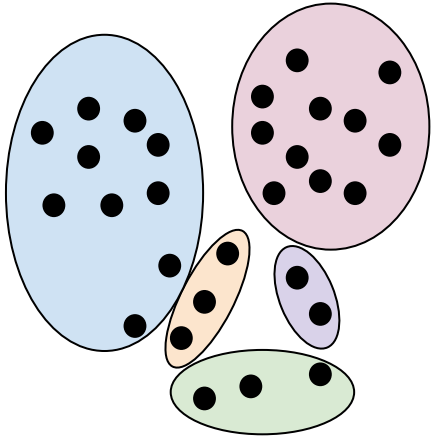
Supervised Learning

Supervised

"We had the worst time at this restaurant. Would never recommend!"

- There is a right answer
- Need annotated label(s) for each sample

Unsupervised



- No labels for data
- Used to explore data
  - Find patterns & groups
- Clustering is popular unsupervised approach

A labelled dataset

created with Flux

- Data quality makes/breaks a machine learning project
  - Garbage in, garbage out
- **Expensive** to prepare as it typically **involves humans**
  - The labels must be assigned accurately and consistently for the classifier to learn a true representation of the task
- Usually a good idea to have multiple assessors label the same items and measure **inter-annotator agreement**.

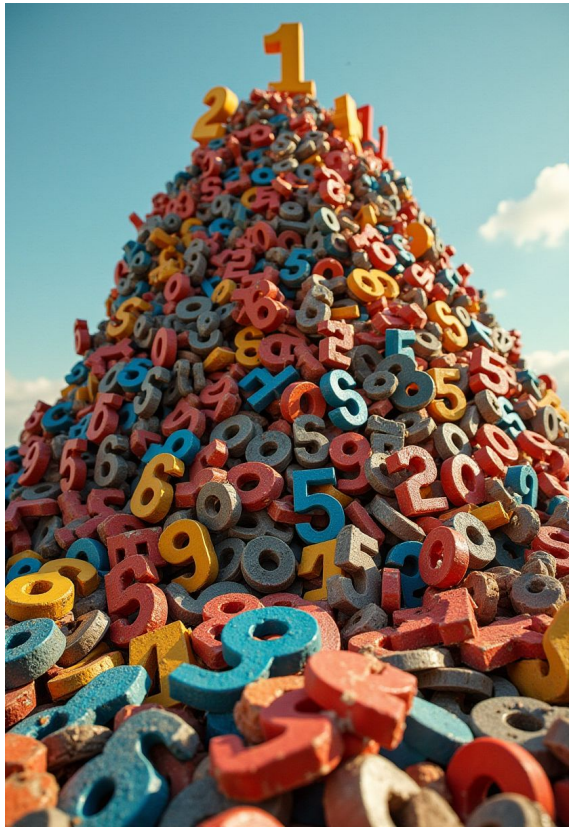
Machine learning algorithms work with numbers

- Numerical
  - -0.73 (floats), 42 (integers)
- Binary
  - TRUE or FALSE / 0 or 1

Often represented with multiple binary values:

- Nominal (i.e. categories)
  - single, married, divorced, etc
- Ordinal (i.e. categories with ordering)
  - small, medium, large

There is no "text" feature type



created with Flux

Representing text with numbers

- Vectorization!!!
- We can represent text as:
  - binary features of word occurrences (c.f. one-hot encoding)
  - continuous features (e.g. TF or TF-IDF) weights

I like **sprite**

Token ID	Token
1	bru
2	i
3	irn
4	like
5	UNK

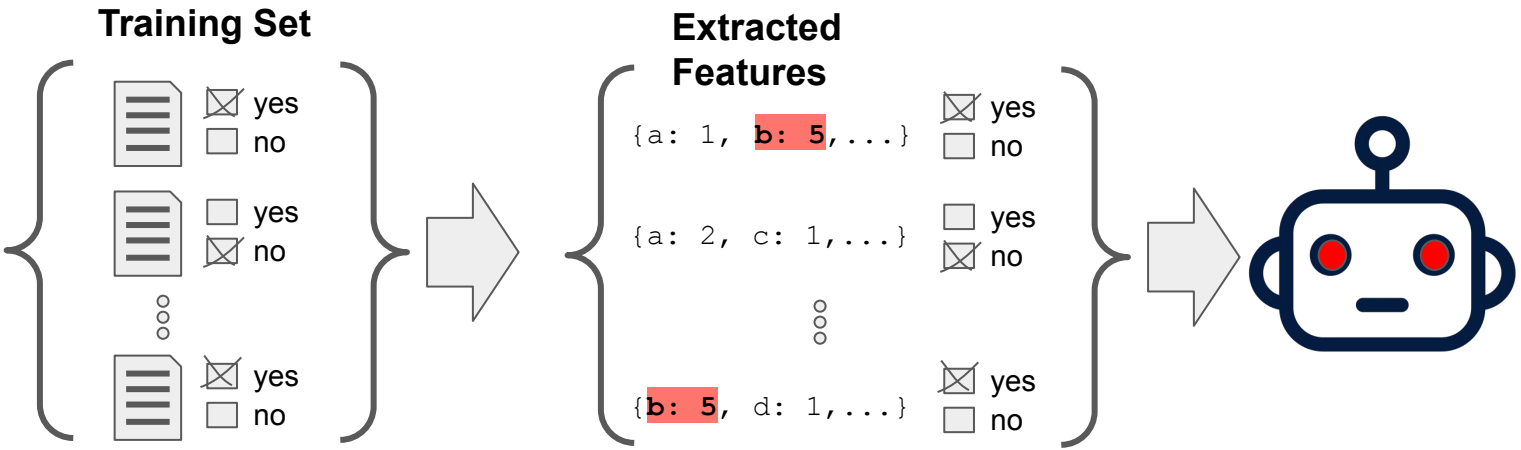
[0,1,0,1,1]

https://scikit-learn.org/stable/modules/feature_extraction.html

Training Process

Features from the input documents are extracted.

The features and classes/labels from **training set** are fed to a machine learning algorithm to train a classifier

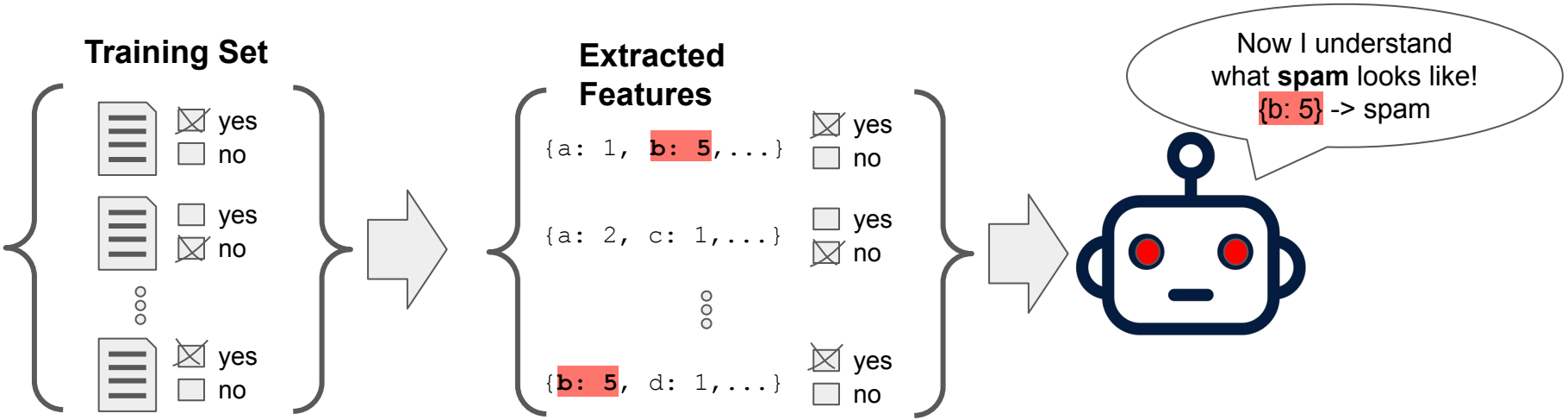


https://scikit-learn.org/stable/supervised_learning.html

Training Process

Features from the input documents are extracted.

The features and classes/labels from **training set** are fed to a machine learning algorithm to train a classifier

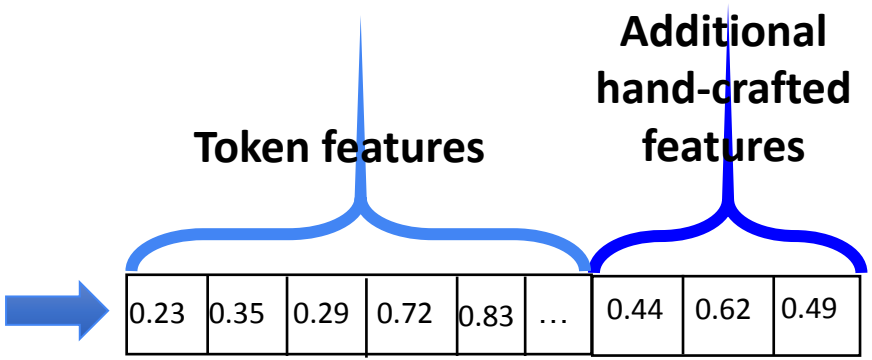


https://scikit-learn.org/stable/supervised_learning.html

Feature engineering

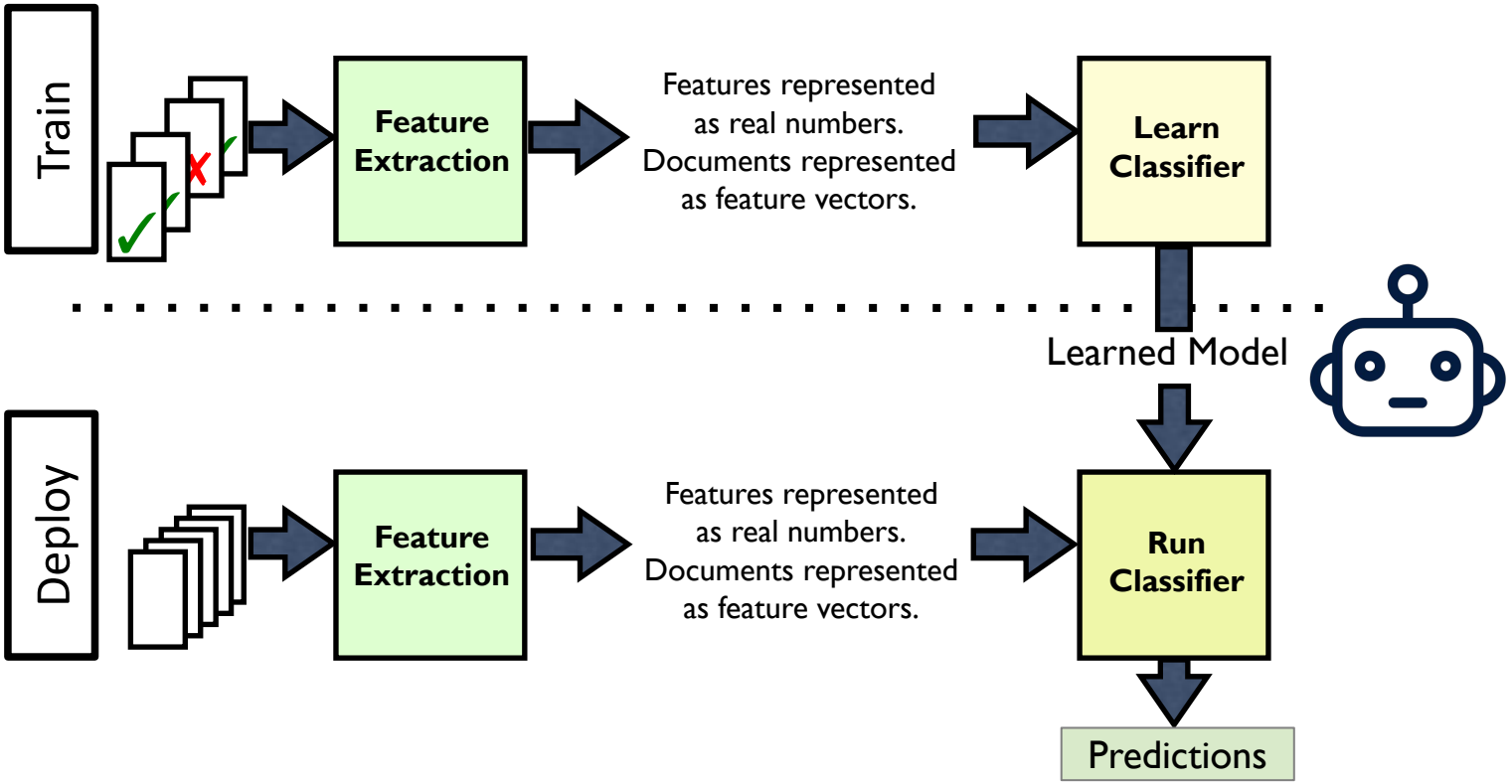
- Some tasks can benefit from hand-crafted features
- Different features can be useful for different tasks
  - Or could confuse a classifier - adding more features is not always useful

All the world's a stage, and all the men and women merely players: they have their exits and their entrances;



*In **author attribution**, average word length, average sentence length and punctuation frequency can be useful additional features (Juola, 2006)*

Supervised Learning: At a Glance

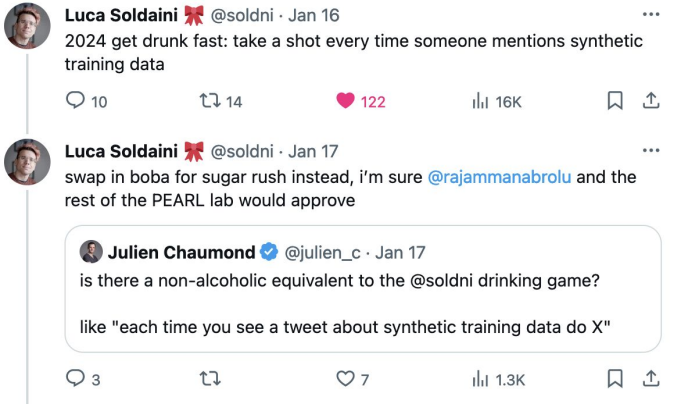


Think-Pair-Share: Synthetic Data

You are a tech investor and considering a new AI company with a classifier product. They aren't going to label their own data. Instead they intend to use ChatGPT to generate synthetic training data

Discuss in your group:

- What are the benefits of using synthetic data?
- What are the risks of using synthetic data?
- Would it affect your investment?



https://padlet.com/jakelever/tad2025_6

Classification Algorithms

A Quick Tour of Classifiers

```
if counts['great'] > 2:
    return 'positive'
else:
    return 'negative'
```

Simple rule-based classifier

- Lots of options of classifiers
  - Advantages and disadvantages to each
  - No definitive best classifier to always use
- Typically learn a “model” from the training data
  - Possibly a set of rules, or some transformation
  - A function (f) that transforms a N-dimensional vector to a label ($\mathcal{L}$)

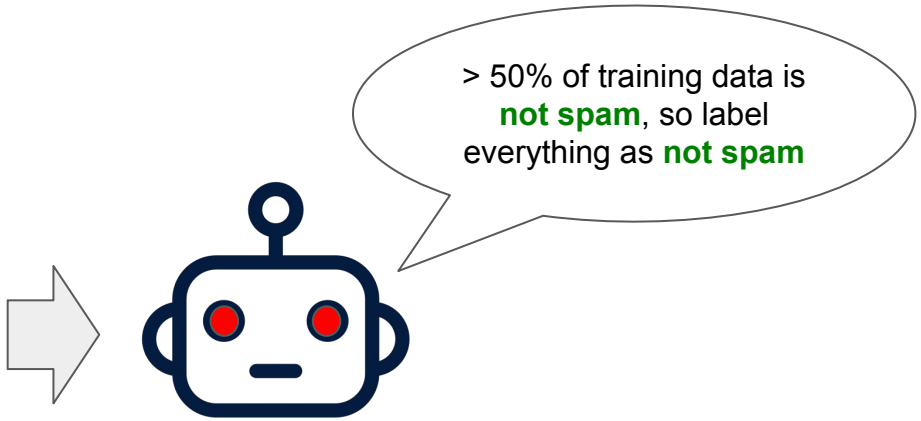
$$f : \mathbb{R}^n \rightarrow \mathcal{L}$$

Majority Classifier (or sometimes ‘Dummy Classifier’)

- Assigns the class that appears most frequently in the training data.
- Does not consider input features, only the labels.
- Doesn’t really “learn” anything – really only useful as a baseline.

Training Data

Not Spam
Not Spam
Spam
Not Spam
Spam
Not Spam
Not Spam



k-Nearest Neighbours (kNN)

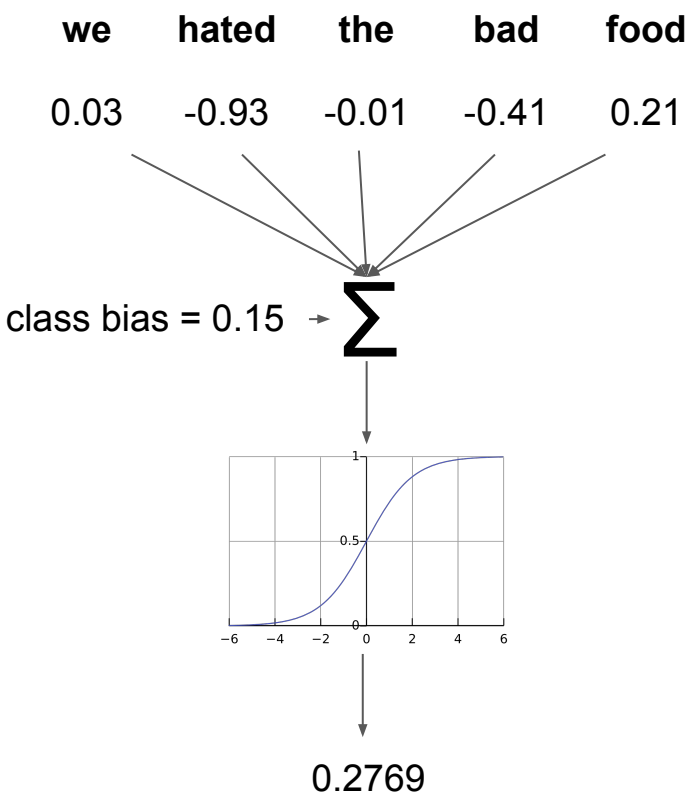
- Method
  - Find the k “nearest” samples from the training set
  - Apply the most common label among those samples
- A lazy learner - no model built
- Expensive - scales badly with the size of the training set

Naïve Bayes

- Applies language modelling using Bayes Rule! (from last lecture)
- Assumes each token is independent (this is why it's called Naïve)
  - This is effectively a unigram model
- Threshold probability at 0.5 to decide if positive/negative

$$P(positive|doc) = \frac{P(doc|positive)P(positive)}{P(doc)}$$

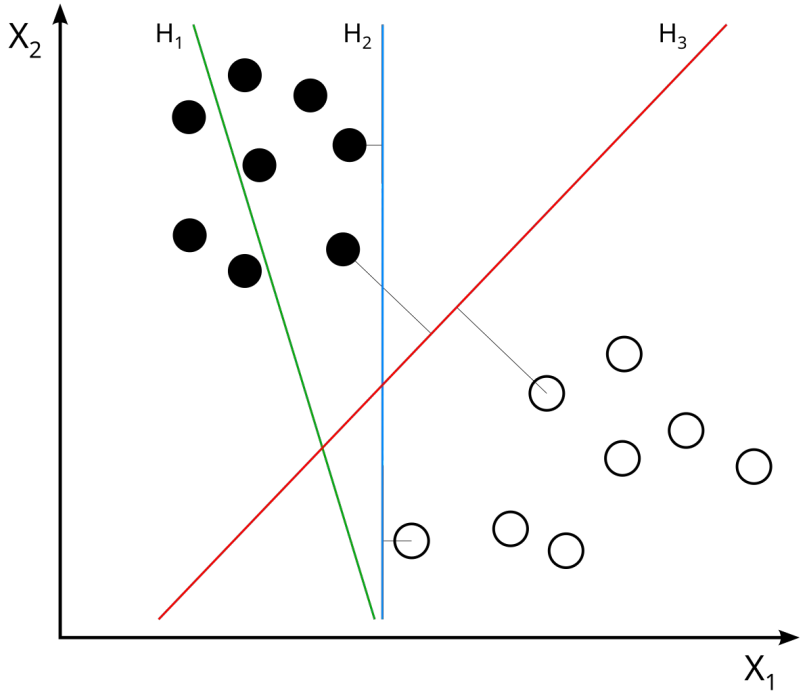
Logistic Regression (LR)



- Assigns a weight to each input feature for each class
  - And a weight (bias term) for each class
- Applies a sigmoid function over the sum of the scores for each class
  - Produces a probability of each class
- Regularisation applied to avoid complex weights.

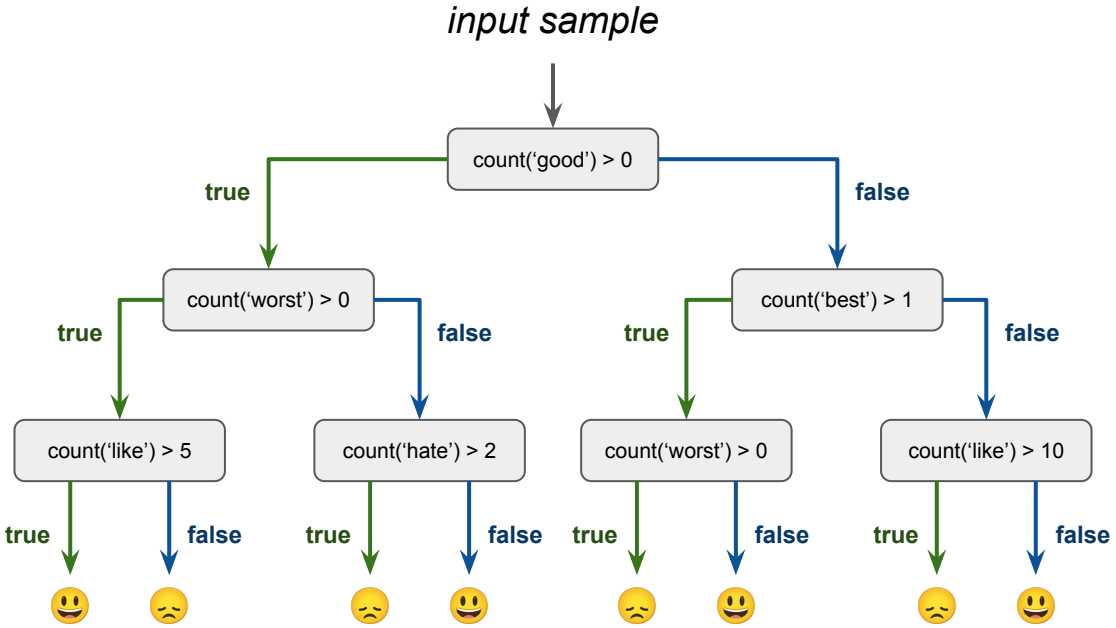
Support Vector Machine (SVM)

- Treat classification as a geometry problem
  - Samples becomes points in an N-dimensional space
- Find the best decision boundary (splitting line) between the two classes
  - Identifies the widest margin between the classes using the training samples (support vectors) along the decision boundary



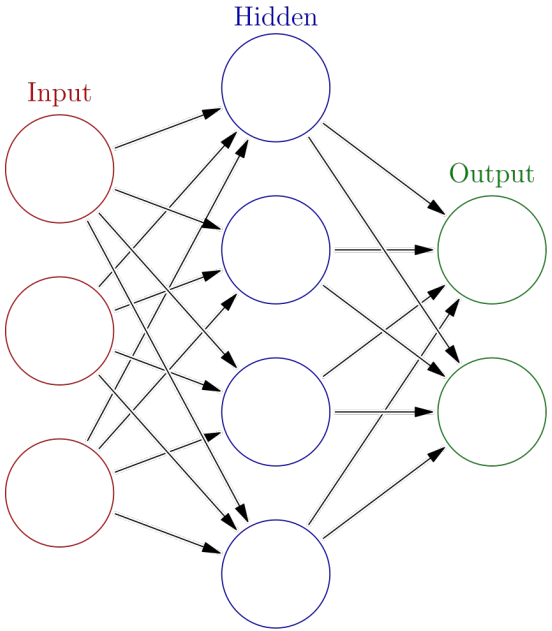
https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg

Decision Tree



- Identifies a set of rules in the form of a tree to make a classification.
- Individual trees are often not very robust, so often multiple trees, each with a random subset of features, are ensembled into a **random forest**

Neural Networks

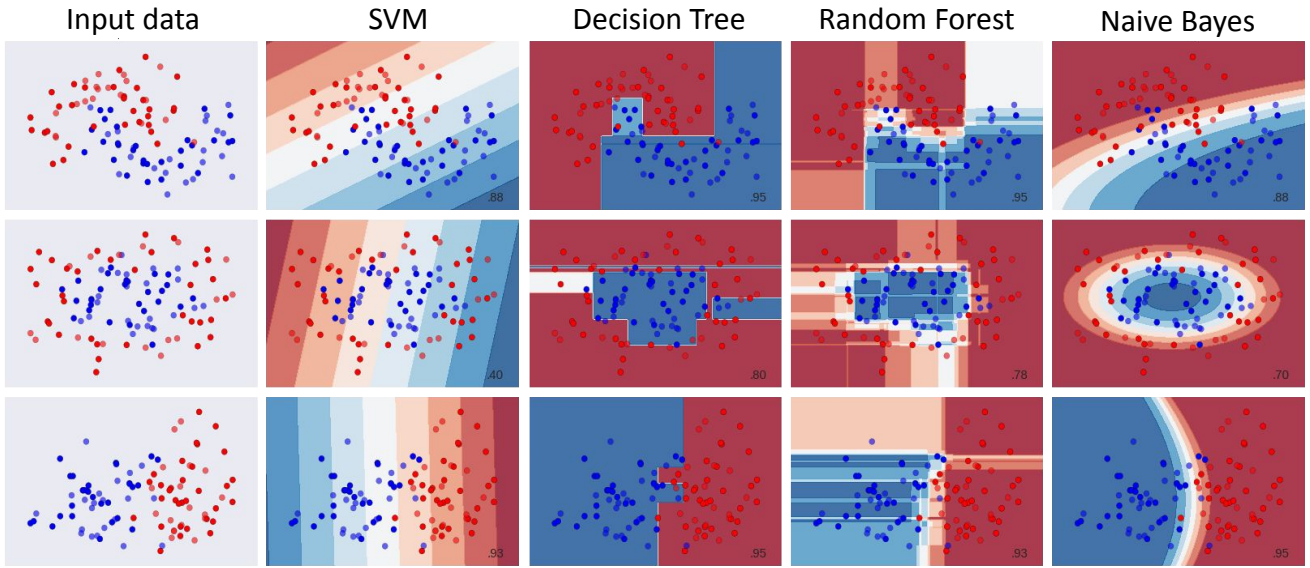


- “Neurons” weight and sum inputs and “fire” when sum is big enough
  - “Firing” is based on an activation function
- Simple network is known are multi-layer perceptron (MLP)
  - Deep learning uses neural networks with LOTS of layers
- Transformers are a neural network architecture - which we’ll go into later

https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Which Classifier to Choose?

- Depends on the case and shape of the data.
- Some will be more appropriate than others



Pros and Cons of different classifiers (for your reference)

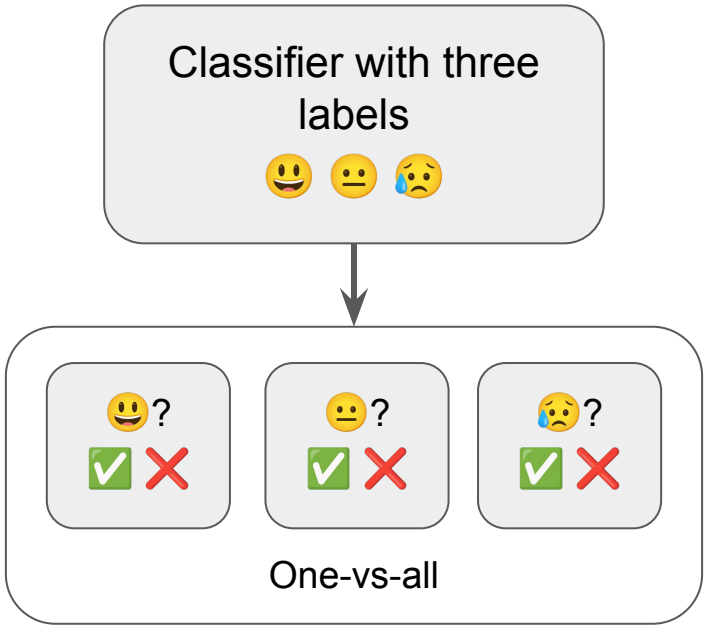
Classifier	Advantages	Disadvantages
Naive Bayes	• Fast & efficient • Probabilistic • Interpretable • Very simple • A relatively effective baseline	• Sensitive to errors on rare words • Weak on rare categories • Weak on very large and noisy documents • Biased due to independence assumptions
Logistic Regression (LR)	• Very efficient • Works with any feature type (no assumptions about distribution) • Widely used baseline for any classification task	• Typically needs more data than NB to train effectively. • Tends to overfit on data without regularisation • Regularisation hyper-parameters need to be fine-tuned using validation data.
Support Vector Machine (SVM)	• Typically more effective than LR for text tasks • Works with any feature type • Few hyper-parameters to tune • Fast on small/medium datasets • Theoretical robustness	• To handle different types of data, need to select a “kernel function” • Slow when scaling to large datasets • Works best when decision boundary is easily separable
Decision Tree	• Fast and cheap • Small trees are easy to interpret • Reasonable accuracy on text tasks • Can handle a variety of input feature types	• Unstable – small change in data leads to very different tree • Less accurate than other methods • Large trees can become complex and difficult to interpret

How to handle multiclass classification

Many Classifiers work only with binary outputs.
We need strategies to combine them

General strategy: Transform the problem to multiple binary classifications

- **One-vs-all (OvA)** □ One binary classifier per class; positive is one class, all others are negative
- **All- vs- All (AvA)** □ discriminate between pairs of classes - (k*k-1)/2 classifiers
- **Issue**: Doesn't scale to large numbers of classes (extreme classification)



https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

Classification Toolkits

Python

Python / C++

Java / Scala

API for Vectorizing in scikit learn

- `fit(X)` Builds a vocabulary on the input data.
- `transform(X)` Creates a document-term matrix (vectorizing)

Be careful with:

- `fit_transform(X)` – performs both in one step
  - only run this once to fit on the training data and then use transform

I like **sprite**

Token ID	Token
1	bru
2	i
3	irn
4	like
5	UNK

[0,1,0,1,1]

Multiple vectorizers

From text

The `sklearn.feature_extraction.text` submodule gathers utilities to build feature vectors from text documents.

`feature_extraction.text.CountVectorizer(*[, ...])`	Convert a collection of text documents to a matrix of token counts.
`feature_extraction.text.HashingVectorizer(*)`	Convert a collection of text documents to a matrix of token occurrences.
`feature_extraction.text.TfidfTransformer(*)`	Transform a count matrix to a normalized tf or tf-idf representation.
`feature_extraction.text.TfidfVectorizer(*[, ...])`	Convert a collection of raw documents to a matrix of TF-IDF features.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text

API for Classifying in scikit learn

- `fit(X, Y)` learns a model, where X is size (N_samples x N_features) and Y is an array of size N_samples
- `predict(X)` applies the learned model to the (unseen) features in X

- Classifiers:
  - GaussianNB
  - LogisticRegression
  - DecisionTreeClassifier & RandomForestClassifier
  - SVC (i.e. SVM), e.g. SVC(kernel="linear") or SVC(gamma=2)
  - Many more!

Hyperparameters

created with Flux

- Classification pipelines often have lots of options to tweak
  - Different vectorization approaches
  - Different classifiers
  - Different classifier settings
- Generally too many to try every possible combination
- Hyperparameter tuning involves trying out a few combinations
  - Evaluating on the validation set

Hyperparameter example

LogisticRegression

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *,
dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None,
solver='lbfgs', max_iter=100, multi_class='deprecated', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[source]

C : *float, default=1.0*

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

- All scikit-learn classifier have a lot of parameters (with reasonable defaults)
- Some are worth playing with (and trying different values)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Hyperparameter tuning in Scikit-learn

- Offers a `GridSearchCV` which can be used to set the parameters of a learner using cross-validation.

- Example: try lots of C values for a few SVM kernels

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = svm.SVC()
clf = GridSearchCV(svc, parameters, cv=3)
clf.fit(X_train, y_train)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Ten minute break

Evaluation

Cross-validation

1. Build your classifier using one set of data
2. Evaluate it on a **different** set of data

- The goal is to understand how your classifier generalises to unseen data

Splitting the dataset

Training set
Used for training a classifier

Validation set
Used to tweak & evaluate lots of different classifiers

Test set
Used for final evaluations

Sometimes known as development set

Common split is 60%/20%/20%

Comparing predictions against the right answer

Email	Actual Label	Predicted Label
Congratulations! You've won a $1,000 gift card. Click here to claim your prize.	Positive	Negative
Hi John, just checking in to see if you're available for lunch tomorrow.	Negative	Positive
Meeting reminder: Team sync-up scheduled for 10 AM tomorrow.	Negative	Negative
You've been selected for a free vacation package! Act now!	Positive	Positive
Get rich quick with this once-in-a-lifetime investment opportunity!	Positive	Positive
Can you please review the attached document and provide feedback by next week?	Negative	Positive

- Is this a good spam classifier?
- Need some numerical metrics to judge success
  - Which metric depends on your task
  - Always a good idea to use multiple metrics

Four possible outcomes (for a binary classifier)

Email	Actual Label	Predicted Label	
Congratulations! You've won a $1,000 gift card. Click here to claim your prize.	Positive	Negative	False Negative
Hi John, just checking in to see if you're available for lunch tomorrow.	Negative	Positive	False Positive
Meeting reminder: Team sync-up scheduled for 10 AM tomorrow.	Negative	Negative	True Negative
You've been selected for a free vacation package! Act now!	Positive	Positive	True Positive
Get rich quick with this once-in-a-lifetime investment opportunity!	Positive	Positive	True Positive
Can you please review the attached document and provide feedback by next week?	Negative	Positive	False Positive

- True Positives & True Negatives: prediction matches correct
- False Positives & False Negatives: mistake!

Confusion Matrix

		Predicted: Positive	Predicted: Negative
Actual	Positive	# of true positives (TP)	# of false negatives (FN)
	Negative	# of false positives (FP)	# of true negatives (TN)

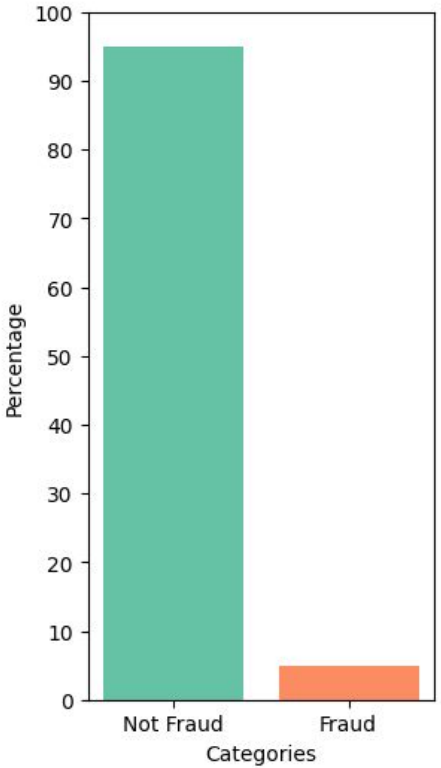
- Grid showing counts of correct and mistaken predictions
  - Also known as contingency table
- Counts used to calculate other metrics
- 2x2 for a binary classification (e.g. spam/not spam)
  - Could be 3x3, 4x4, etc for multi-class classification problems

Accuracy - % of predictions that match actual label

Actual Label	Predicted Label	
Positive	Negative	False Negative
Negative	Positive	False Positive
Negative	Negative	True Negative
Positive	Positive	True Positive
Positive	Positive	True Positive
Negative	Positive	False Positive

$$accuracy = \frac{\#correct}{\#samples} = \frac{TP + TN}{TP + TN + FP + FN}$$

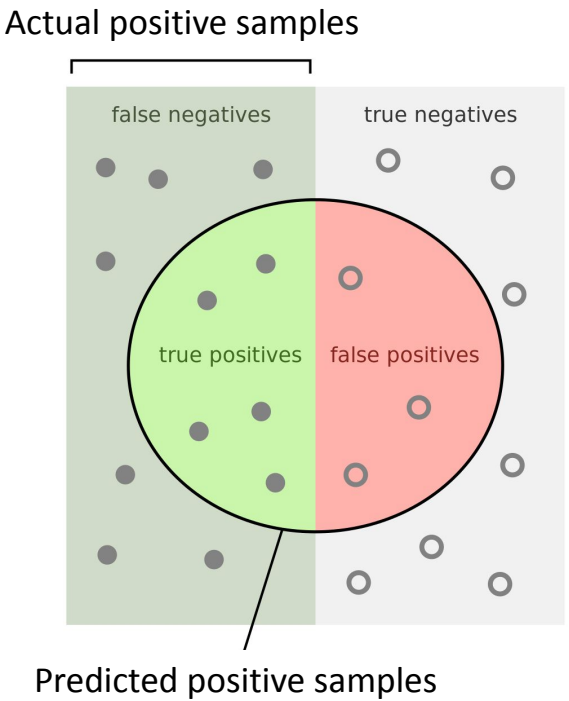
The Problem with Accuracy



Class Imbalance Problem:

- One class may be rare, e.g. fraud, or has-disease
- Easy way to get high accuracy in such cases: always predict the majority class

Precision

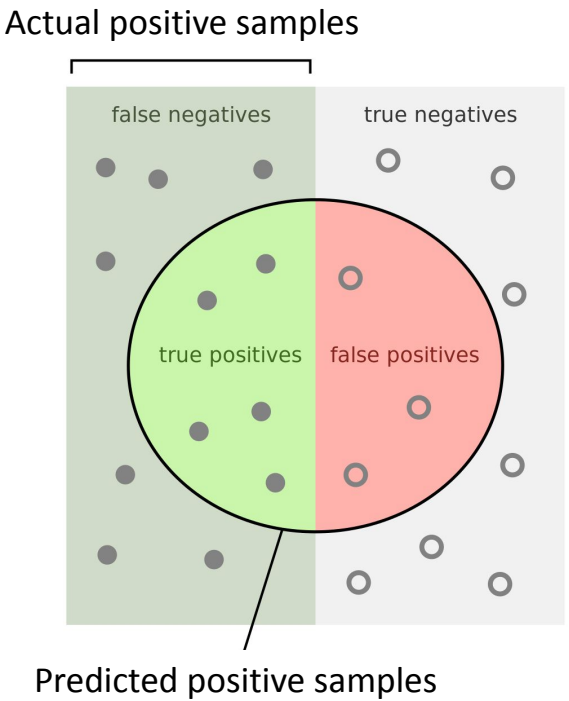


$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}} = \frac{TP}{TP + FP}$$

What fraction of the predicted positives are actually positive?

https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

Recall



$$\text{Recall} = \frac{\text{●}}{\text{●}} = \frac{TP}{TP + FN}$$

What fraction of the predicted positives are actually positive?

Also known as sensitivity and the true positive rate

https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

F_1 score

- Precision and recall are both important
- Let's create a single score that is the **harmonic mean** of them

$$F_1 = 2\frac{precision \times recall}{precision + recall}$$

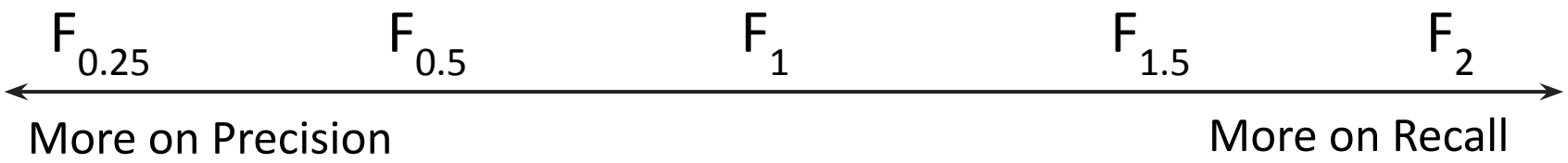
$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$

The most common metric used in all of machine learning

The generalised F_β score

- Do you care more about false positives or false negatives?
  - Depends on your task
- F_β score allows you to adjust the balance of precision and recall

$$F_\beta = (1 + \beta^2) \frac{precision \times recall}{\beta^2 \times precision + recall}$$



Does the F_1 score capture everything?

$$F_1 = 2\frac{precision \times recall}{precision + recall}$$

$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$

		Predicted: Positive	Predicted: Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- F_1 is very popular but doesn't include every aspect of performance
- **No metric captures all aspects of performance!**

Summary of main metrics

$$accuracy = \frac{\#correct}{\#samples} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = 2\frac{precision \times recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

		Predicted: Positive	Predicted: Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Multi-class problems

- No clear positives and negatives
  - So no TP, FP, etc counts
  - Can't directly calculate precision, recall & F_1
- But we can do accuracy!

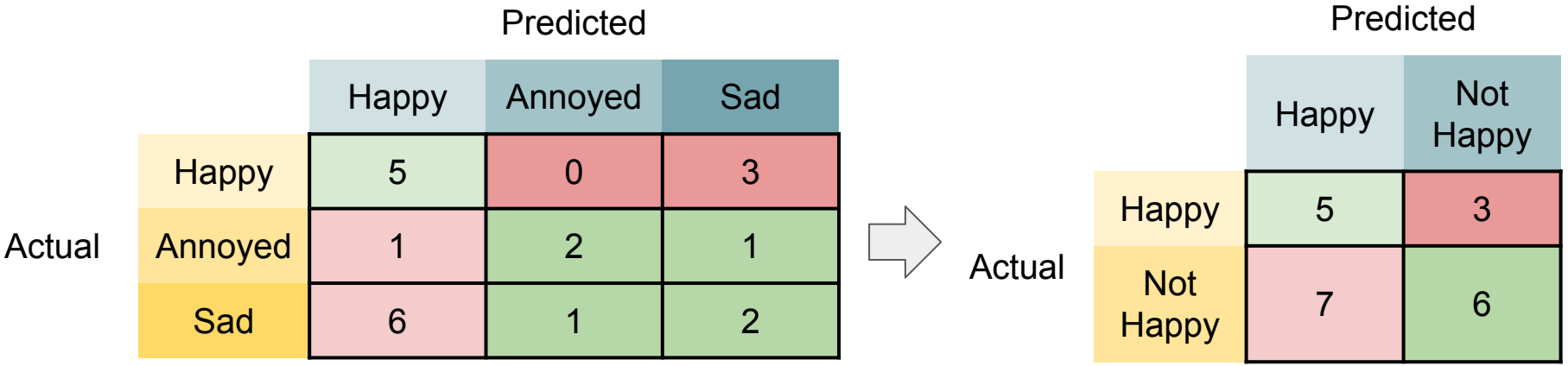
		Predicted		
		Happy	Annoyed	Sad
Actual	Happy	5	0	3
	Annoyed	1	2	1
	Sad	6	1	2

Accuracy for a multi-class problem

		Predicted		
		Happy	Annoyed	Sad
Actual	Happy	5	0	3
	Annoyed	1	2	1
	Sad	6	1	2

$$accuracy = \frac{\#correct}{\#samples}$$

Precision / recall / F1 for multi-class problems



- Make a binary confusion matrix for each class (class & not class)
- Calculate per-class precision, recall, F1-score

Macro average

	Precision	Recall	F_1 score	# actual samples
Happy	0.42	0.62	0.50	8
Annoyed	0.67	0.50	0.57	4
Sad	0.33	0.22	0.27	9
Macro average	0.47	0.45	0.45	21

- Get per-class precision recall and F1 score
- Calculate the unweighted mean of them
  - Doesn't factor in class balance (which may be an advantage!)

Two averaging methods that factors in class balance

Weighted average: Use counts in the average calculation

$$\frac{\#_{happy} \times precision_{happy} + \#_{annoyed} \times precision_{annoyed} + \#_{sad} \times precision_{sad}}{\#_{happy} + \#_{annoyed} + \#_{sad}}$$

$$TP = TP_{happy} + TP_{annoyed} + TP_{sad}$$
$$FP = FP_{happy} + FP_{annoyed} + FP_{sad}$$
$$FN = FN_{happy} + FN_{annoyed} + FN_{sad}$$
$$TN = TN_{happy} + TN_{annoyed} + TN_{sad}$$

Micro average: Calculate global TP, FP, FN and TN counts from per-class confusion matrices

Evaluating multi-label classifications

Document	Gold Labels	Predicted Labels
"AI Breakthrough: New Algorithm Promises Faster Disease Diagnosis"	Tech, Health	Tech, Business
"Tech Giant Acquires Startup in $2 Billion Deal"	Tech, Business	Tech
"Stock Market Surges as Retailers Report Record Earnings"	Business	Business
"How Blockchain is Revolutionizing Global Supply Chains"	Tech, Business	Tech, Business
"Oil Prices Surge as OPEC Announces Unexpected Production Cuts"	Business	Tech, Business
"New Smartphone Model Sets Benchmark for Battery Life"	Tech	Tech, Health

- Cannot create one single confusion matrix
  - No accuracy measure
- Create a confusion matrix for each label
  - Calculate precision, recall, etc
  - Use averages (e.g. macro-F_1)

Most classifiers actually give scores, not just output labels

Input: Delicious food. Would go again
Positive Class Score: 0.82

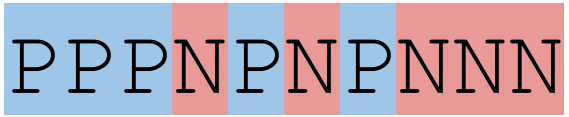
Input: Horrible place. Never again
Positive Class Score: 0.05

Input: Bad food. Great staff
Positive Class Score: 0.47

- Many classifiers produce a score for each class
  - Not just a predicted label
- Can choose a threshold score
  - Allows trade off of precision and recall
- How would you evaluate these scores directly?

Ideal case for scores

Actual Label	Score from Classifier
Positive	0.946
Positive	0.835
Positive	0.650
Negative	0.563
Positive	0.359
Negative	0.327
Positive	0.290
Negative	0.167
Negative	0.070
Negative	0.027

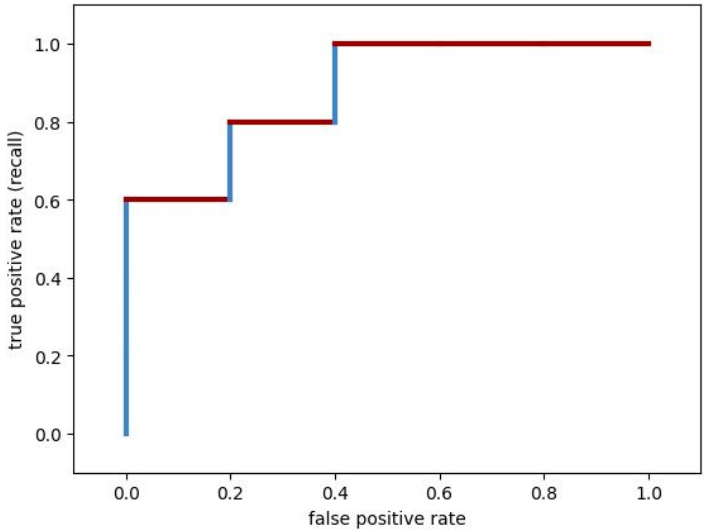


- High scores for positives
- Low scores for negatives
- Order matters (actual scores not really)

We'd like something more quantitative

Receiver Operating Characteristic (ROC) curve

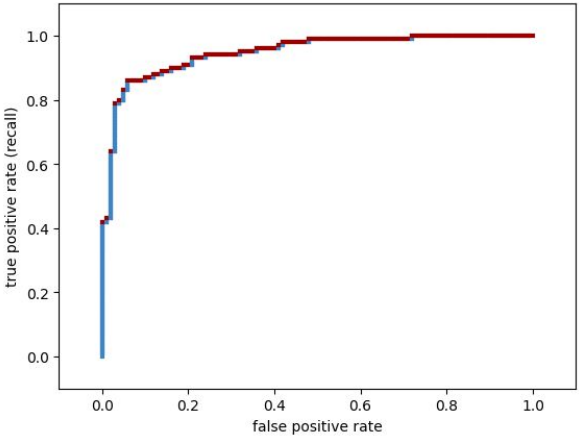
PPPNPNPNNN



- Start in the bottom left (and at the start of your ordered list of predictions)
  - Move up for every Actual Positive and right for every Actual Negative
- Effectively thresholding the scores at each value, deciding positive/negative predictions and calculating recall (TPR) and FPR

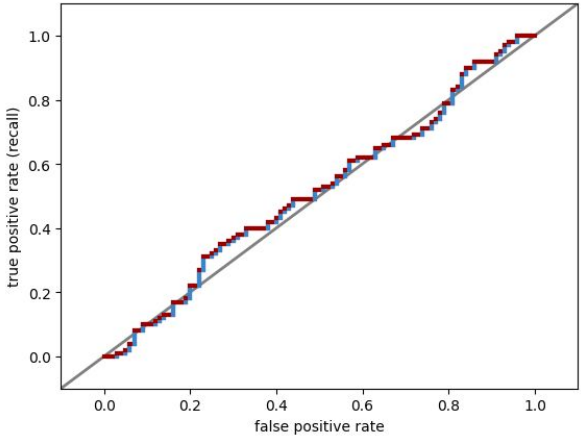
$$FPR = \frac{FP}{FP + TN}$$

Good and bad ROC curves



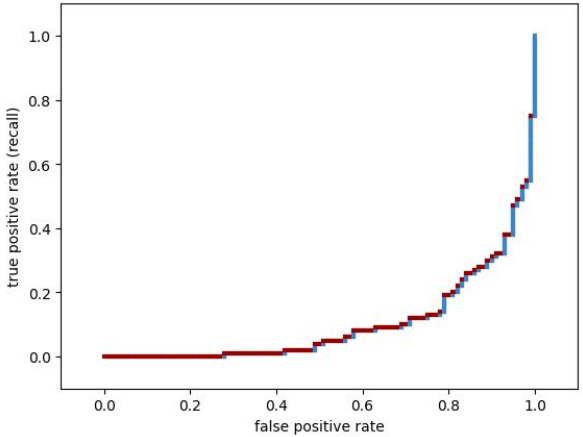
Good

- Actual positives have highest scores



Random (bad!)

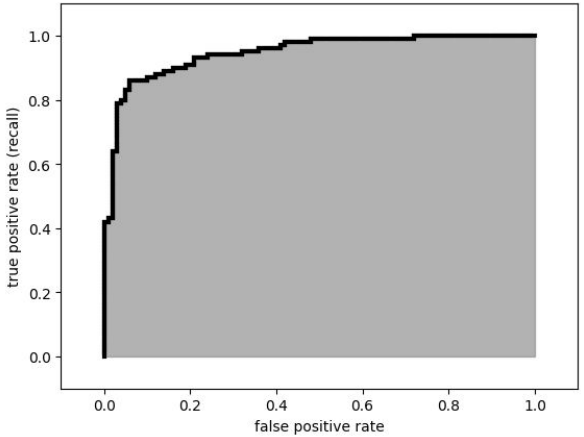
- Similar scores for positives & negatives
- Close to y=x



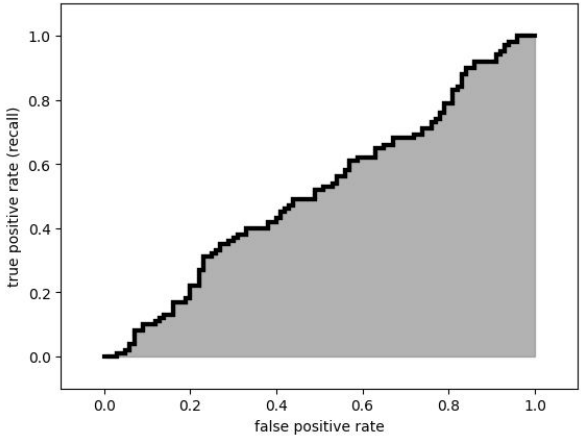
Disaster (bad!!!)

- Actual negatives have highest scores

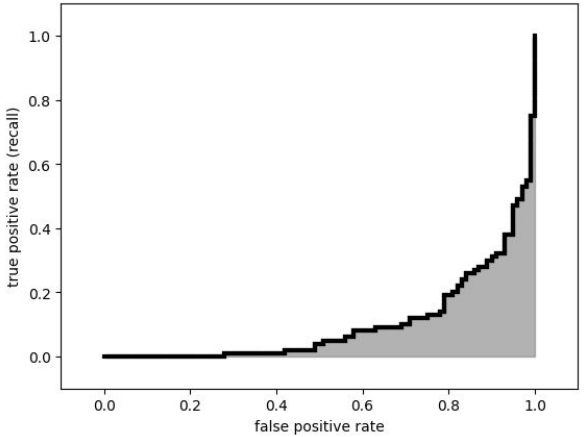
Area under the Curve (AUC) of a ROC curve



AUC = 0.947



AUC = 0.514



AUC = 0.101

- Area of the shaded area under an ROC curve
- Range from 0.0 to 1.0 with higher being better
- 0.5 is a random classifier

Which model is better?

Classifier	F1-score
Model A	0.682
Model B	0.688
Model C	0.684

Evaluated on test set with N=27

Voting choices:

1. Model A
2. Model B
3. Model C
4. Eh, maybe they're about the same?
5. What do you mean by better?

Statistical tests

- Higher scores != better
- Small differences could be down to random chance
  - e.g. the model happens to be slightly better at this test set, but not representative that it would always be better in real life
  - Very dependent on the size of the test set
- Can use statistical tests to check if the improvement is statistically significant

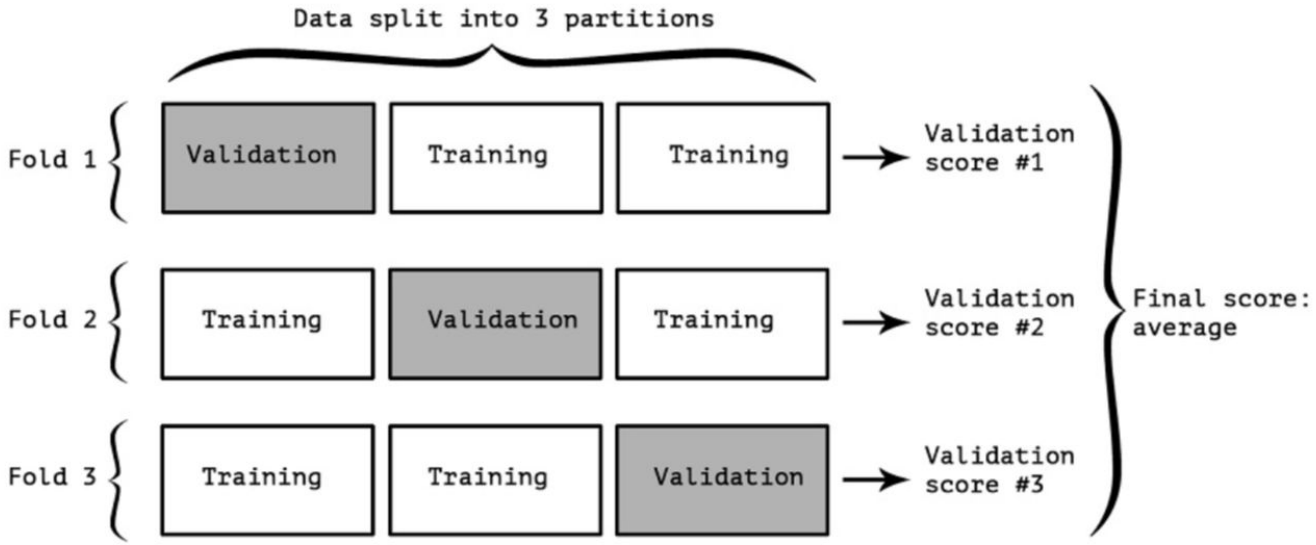
Classifier	F1-score
Model A	0.682
Model B	0.688
Model C	0.684

Evaluated on test set with N=27

Summary of Metrics for Binary, Multi-class & Multi-label

Type	E.g. Tasks	Classifiers	Metrics
Binary	spam vs not spam, positive vs negative sentiment	All classifiers can be used.	Precision(P), Recall (R), $F_{1,}$ AUC/ROC
Multi-class	language identification, author attribution (single author)	Naïve Bayes k-NN Decision trees	Accuracy, micro or macro averaged P, R, F_1
Multi-label	image content, author attribution (multiple authors)	Multiple binary classifications	micro or macro averaged P, R, F_1

K-Fold Cross-validation (for small datasets)



- Partition the data into k mutually exclusive subsets
- Measure performance (e.g. F1 score) on different folds and average
- You'll still want a separate test set!

Summary

- Introduced the task of text classification
  - Representing text as features for ML algorithms
  - Defined the different types of classification tasks

- Examined key types of supervised text classifiers
  - Binary and multiclass algorithms
  - Important methods: Naïve Bayes, Logistic Regression, SVM, Trees, Neural networks

- Evaluation measures for classification tasks
  - Precision, recall, F1, AUC

Breather

Building a good classifier

Why doesn’t my classifier do better?

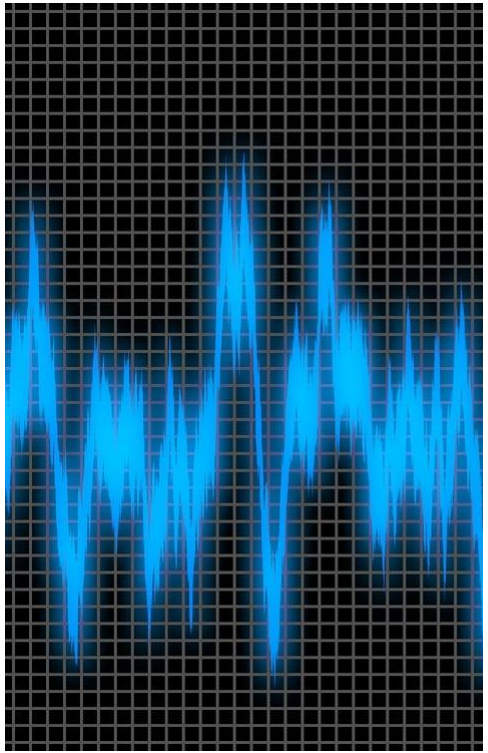
Problems with…

- the data?
- the features?
- the classifier?

Noise in the training data

- **Label noise**
  - human assessors were not clear on the task
- **Feature noise**
  - In sentiment analysis, someone had a typo in their review ("I'm [not] happy ☹️")

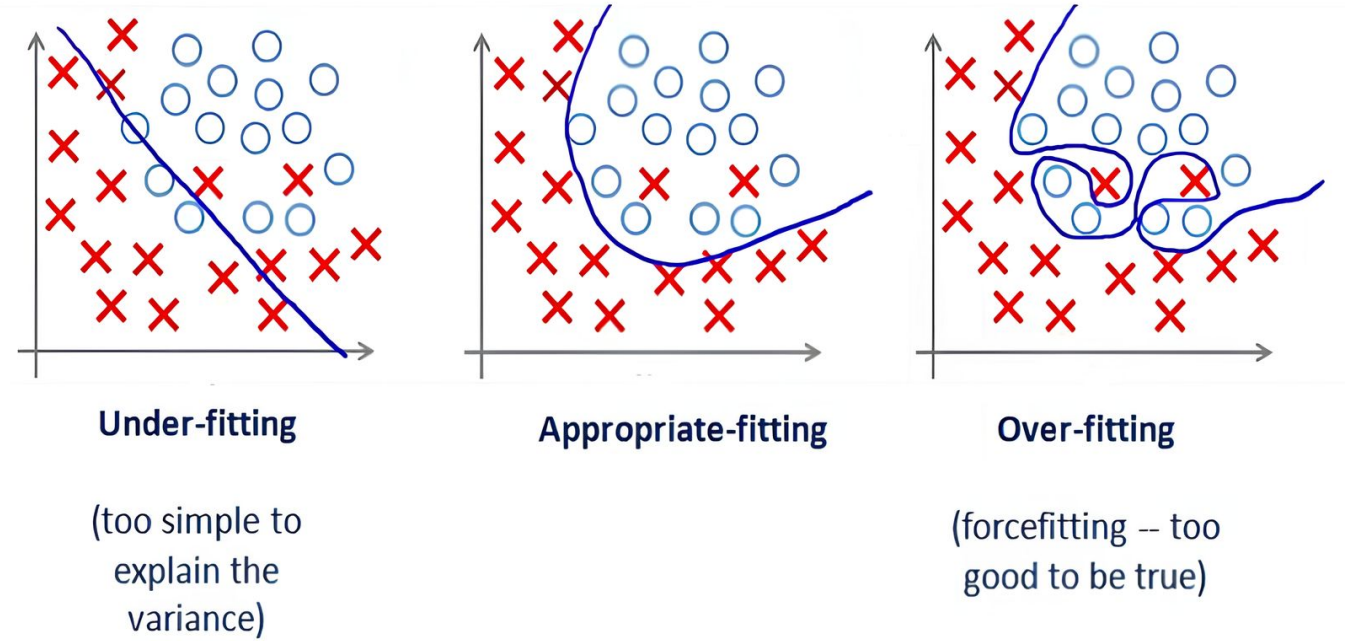
Data cleaning (or working with the data providers) may help with these issues



The needed features are just not there

- Features might not be sufficient for learning:
  - In a medical context, you might have the patient's history, genetic data, X-rays, family histories. Lots of data!
  - But it may not be enough to predict if that patient has early stage cancer or not

Underfitting vs Overfitting



Underfitting occurs when the model cannot capture the characteristics of the problem
Overfitting occurs when a model **too closely fits** to a **limited set** of data points.

Dealing with over/underfitting

created with Copilot / DALL-E 3

- Try different classifiers
  - Some classifiers (e.g. neural networks) are more complex and can fit more complex patterns (but may overfit more easily)
- Hyperparameter tune!
  - Some hyperparameters are there specifically for this
  - Regularization controls the “complexity” of a model

Classifiers don’t always use the patterns we want/expect

Text	Label
The service was **impeccable**, and every dish was bursting with flavor.	Positive
I waited over an hour for a **cold** meal and a **half-hearted** **apology**.	Negative
A long wait, **rude** staff, and **burnt** food made for a **terrible** night out.	Negative
Portions were tiny for such a high price, making it a complete **rip-off**.	Negative
The seafood was so **fresh** it tasted like it came straight from the ocean.	Positive
From the first sip of wine to the last bite of dessert, everything was **perfection**.	Positive

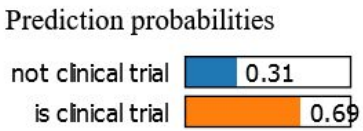
What we hope is happening

Text	Label
The service was impeccable, and every dish was bursting with flavor.	Positive
I waited over an hour for **a** cold meal and **a** half-hearted apology.	Negative
A long wait, rude staff, and burnt food made for **a** terrible night out.	Negative
Portions were tiny for such **a** high price, making it **a** complete rip-off.	Negative
The seafood was so fresh it tasted like it came straight from **the** ocean.	Positive
From **the** first sip of wine to **the** last bite of dessert, everything was perfection.	Positive

What may be happening

Interpretability and explainability may help

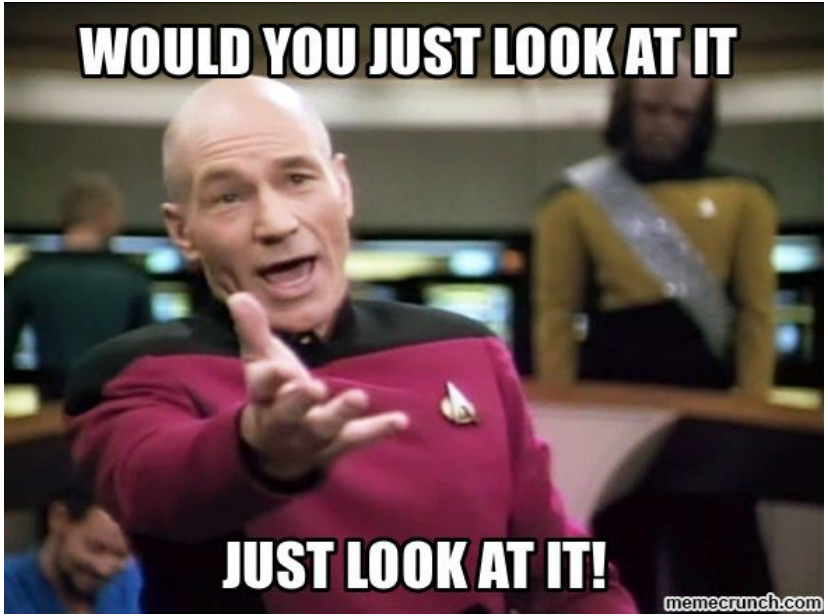
- Some classifiers (e.g. logistic regression) can be easily inspected
- Methods such as LIME/SHAP can be used to visualise what features are being used to make predictions
- Explainability is a HARD problem
  - And often involve vast simplification of what a classifier is actually doing



Text with highlighted words

A clinical trial of the bronchodilator effect of Sch 1000 aerosol in asthmatic children.
In a double-blind study in 23 asthmatic patients Sch 1000 was found to be an effective bronchodilator with an onset of effect within 15 minutes and a duration of four hours. It was effective on both small and large airways.

Error analysis



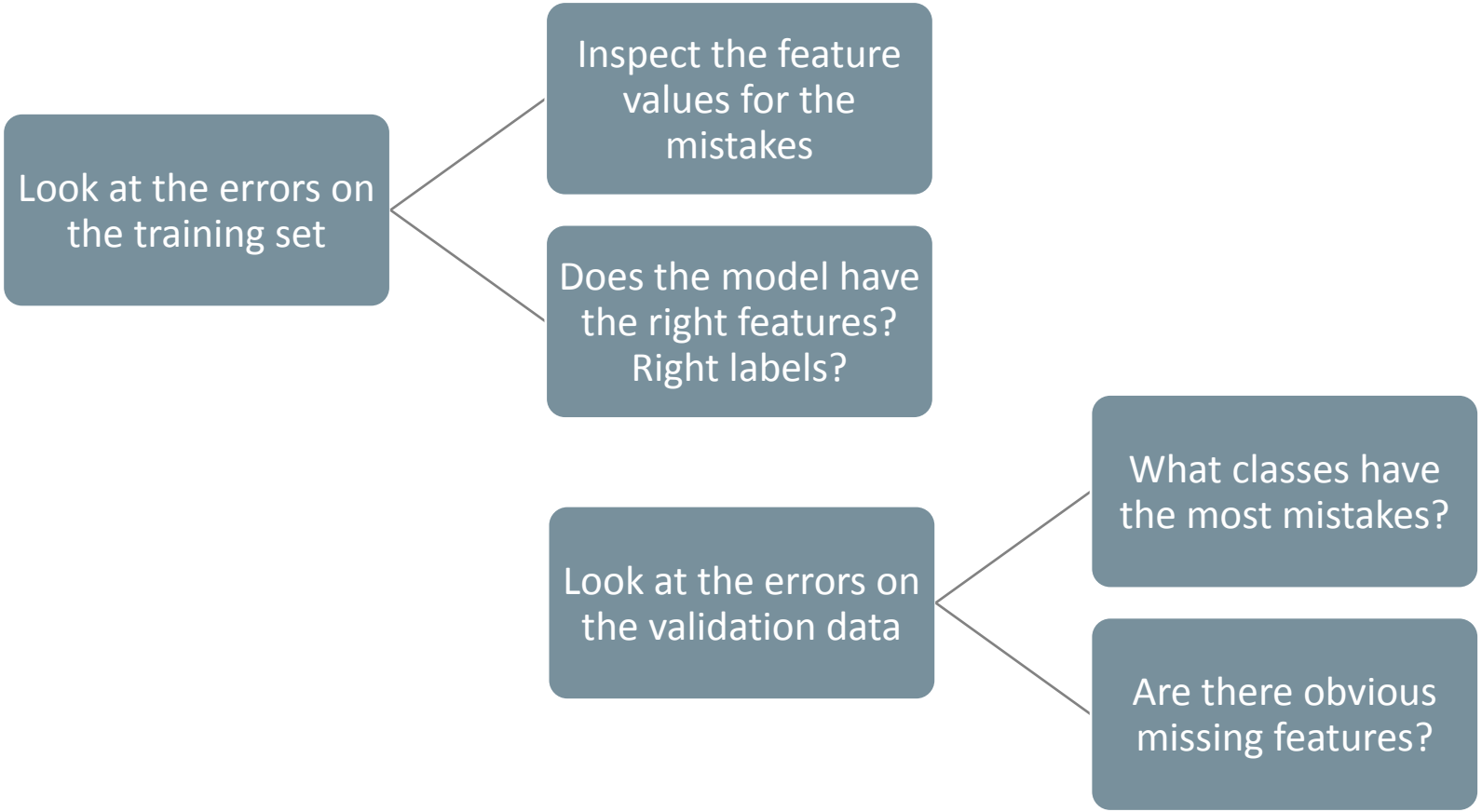
What is your classifier good and bad at?

Essential knowledge for building and deploying an ML system

Are there any obvious biases or other problems?

- Still refining your model?
  - Look at mistakes in your training & validation datasets
- Finished and want some final conclusions?
  - Can look at mistakes in test set
  - **BAD** to use this knowledge to refine your model

Error analysis to improve your model



Error analysis example

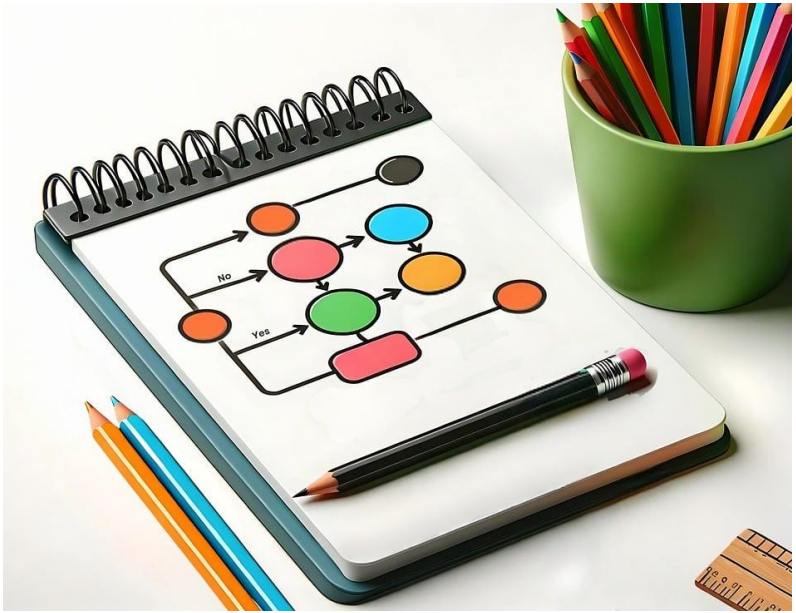
created with Copilot / DALL-E 3

- You build a classifier for predicting the topic of a news article
- Someone says your classifier is misclassifying sports content as business

Is that true? Should you invest time in solving it?

Does it matter that it misclassifiers sports & business?

- Get ~100 mislabeled examples from the validation set
- Count up how many are about sports
- If you only have 5% sports
  - Solving this problem will have a minor impact
  - Don’t waste your time
- If you have 50% sports,
  - Solving this problem will have a big impact on overall performance!



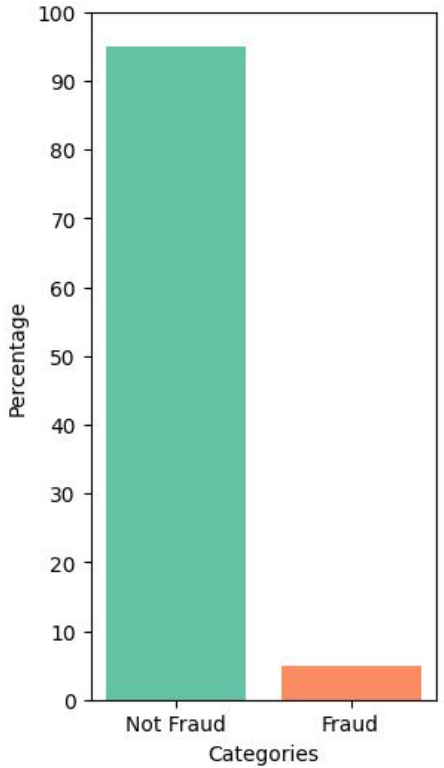
Other tricks for tweaking performance

- Usually very task-dependent. Look at and understand your data!
- Domain-specific features and weights: very important in real performance
  - May need domain expertise to create useful features
  - Sometimes need to collapse terms:
    - Parts numbers, chemical formulas, …
- Upweighting: Counting a word as if it occurred twice:
  - title words (Cohen & Singer 1996)
  - first sentence of each paragraph (Murata, 1999)
  - In sentences that contain title words (Ko et al, 2002)

Class Imbalance

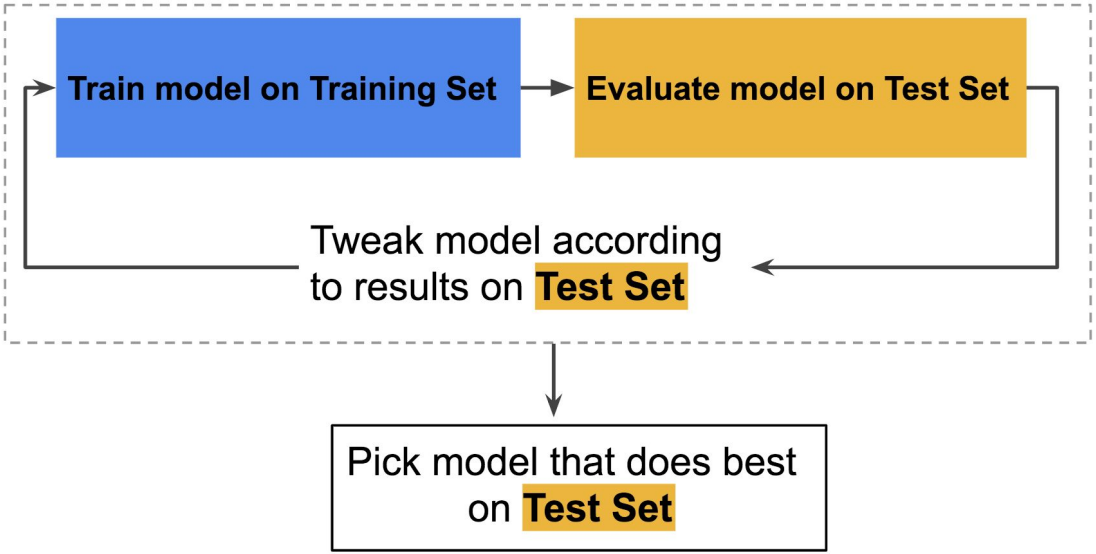
Many real-world problems have substantial class imbalances

1. Be careful with evaluation
   - Accuracy can easily be tricked
2. Training classifiers may also be trickier
   - Poor training may lead to a classifier that only predicts the majority class
   - Some classifiers can take class imbalance into account while training
   - Resampling the training set to an equal balance may also help



The Right Workflow

What's wrong with this workflow?



Your test data is sacred!

The goal

a classifier that works well on unseen data

not a classifier that works well on your specific data

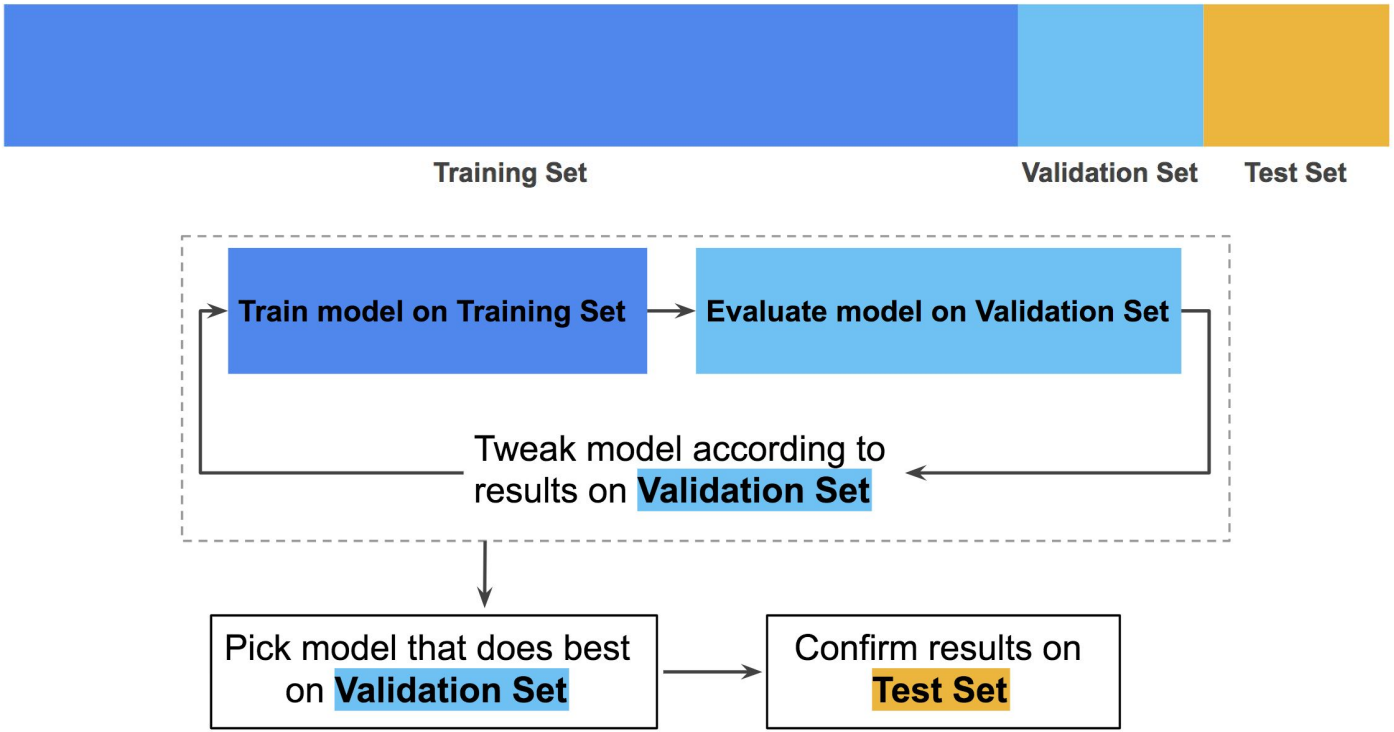
Using the test set (in any way) to build your classifier **undermines** this

Train/Validation/Test *splits*

- **Mistake 1:** We do **NOT** use all of our data for training.
- **Mistake 2:** We do **NOT** measure (final) performance on training data.

- We split the data into training, validation, test
  - Training: for model construction
  - Validation: for setting hyperparameters, error analysis (sometimes called “dev”)
  - Testing: for performance estimation

The Right Workflow



If you do it right your model should…

Fit training set well

Fit validation set well

Fit test set well

Perform well in real world!

Summary of Text Classification

- Classification Task: Binary, Multi-Class & Multi-Label
- Classifiers
  - Lots of different methods (but similar code in scikit-learn)
- Evaluation
  - Cross-validation
  - Confusion matrix
  - Accuracy & Precision, Recall, F1-score
- Improving your classifier
  - Look at your data!
- Respect your test data!

Next Week's Lab & Office Hours changed this week

- Next Lab: Text Classification
- Office Hours: Friday 2pm-3pm (SAWB 331)