Text Classification

Text Mining, Transforming Text into Knowledge

Ayoub Bagheri

Last week

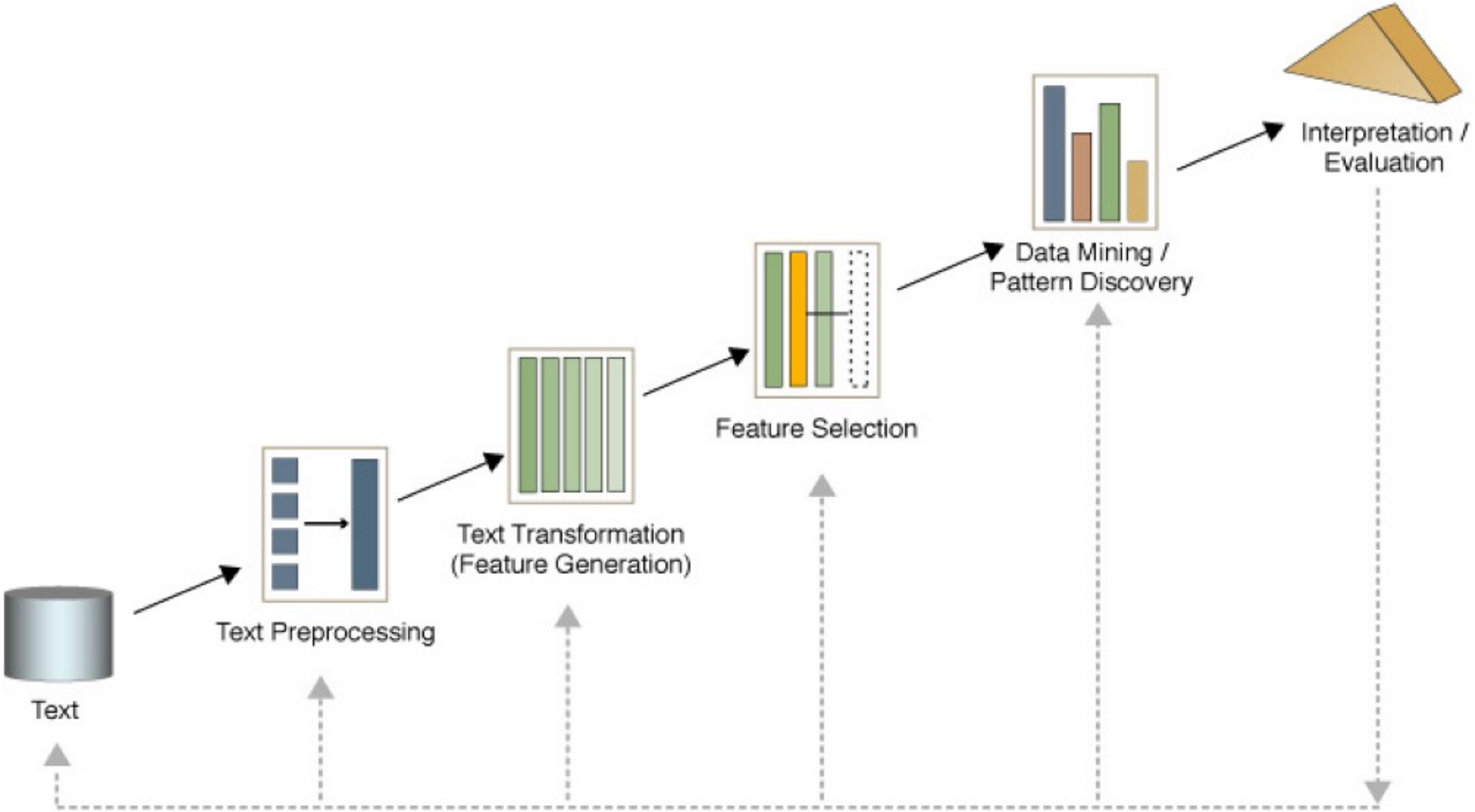
- Text preprocessing
- Tokenization
- Token removal
- Stemming
- Text representation

Today

- Text classification
- Classification pipeline
- Classification algorithms
- Evaluation

Text Classification

Text mining process



Text mining process

- **Data: Text**
- **Text Preprocessing:** is the process of cleaning, normalizing, and structuring raw text data into a format suitable for analysis or input into NLP models. (week 2)
- **Text transformation, feature generation:** involves converting text data into a different format or structure, such as numerical vectors or simplified forms, to make it suitable for analysis or modeling. (weeks 1, 2, 3, 6, 7, 8)
- **Feature selection**: is the process of identifying and selecting the most relevant features from a dataset to improve model performance and reduce complexity. (week 4)
- **Data mining, pattern discovery**: is the process of extracting meaningful patterns and knowledge from text. (weeks 3, 5, 7, 8, 9)
- **Interpretation / Evaluation:** is the process of understanding and explaining the model and patterns / is the assessment process to measure performance and quality. (weeks 3-9)

Text transformation, feature generation

Bag-of-Words representation

Doc1: Text mining is to identify useful information.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

Document-Term matrix (DTM):

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious
Doc1	1	1	1	1	0	1	1	1	0	0	0
Doc2	1	1	0	0	1	1	1	0	1	0	0
Doc3	0	0	0	0	0	1	0	0	0	1	1

Text classification

- **Supervised learning**: Learning a function that maps an input to an output based on example input-output pairs.
  - infer a function from labeled training data
  - use the inferred function to label new instances

- Human experts annotate a set of text data
  - Training set

Document	Class
Email1.....	Not spam
Email2....	Not spam
Email3....	Spam
...	...

Applications

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis

Quiz?

- Which one **is not** a text classification task? (less likely to be)
  - Author's gender detection from text
  - Finding about the smoking conditions of patients from clinical letters
  - Grouping similar news articles
  - Classifying reviews into positive and negative sentiment

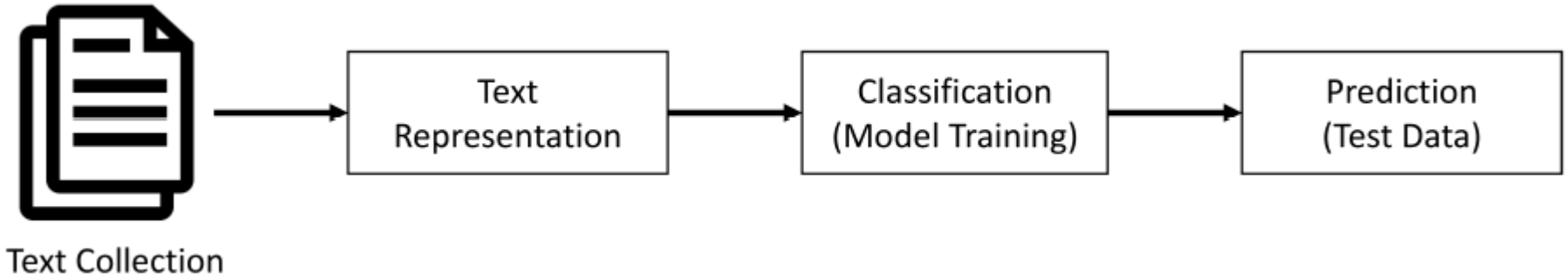
Terminology

- **Features**: information which is used to separate data into classes
- **Prediction**: the output of the model, which can be compared to the correct labels (our ground truth)
- **Parameters**: the data in the model
- **Training**: fitting parameters to the data
- **Hyperparameters**: the “settings” of the model; these are choices made by you

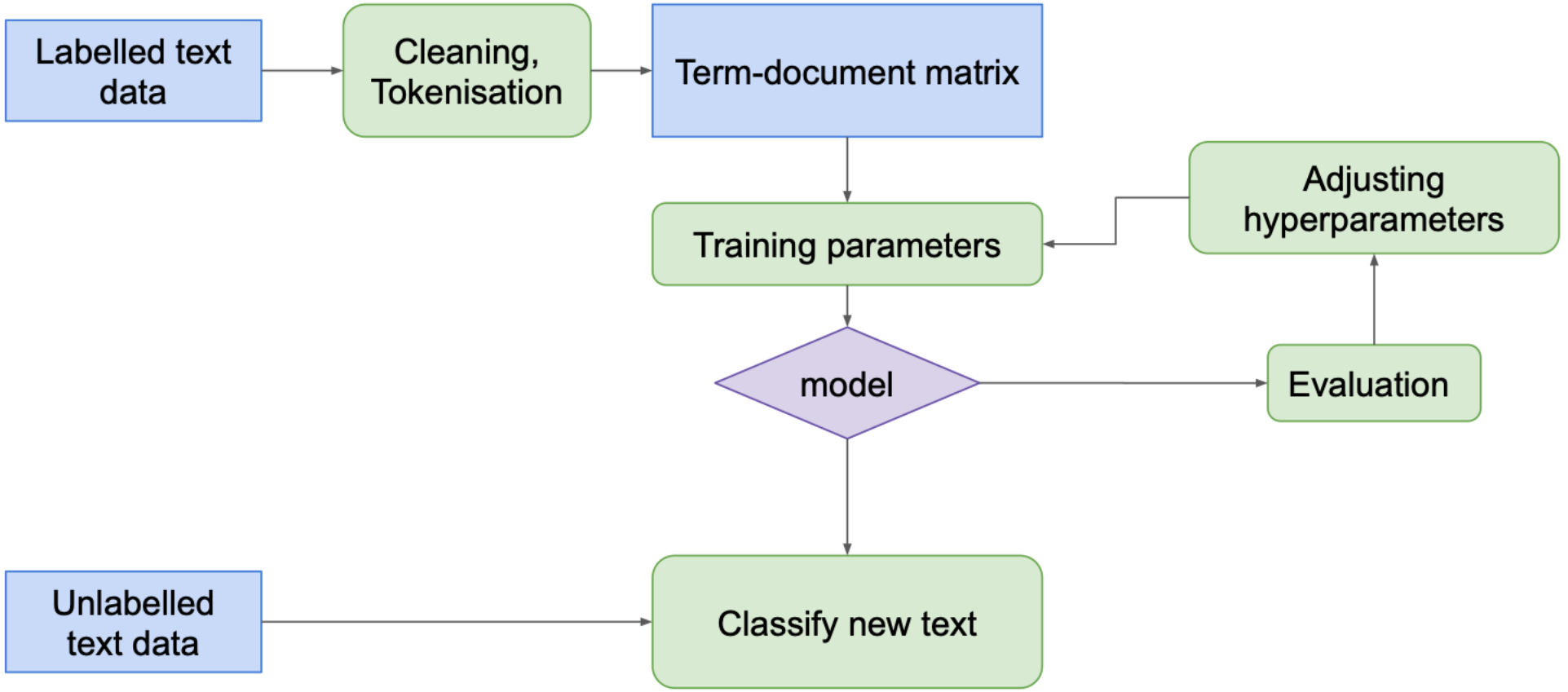
Types of classification

1. **Binary classification**:
   - **Definition**: Involves two categories or classes.
   - **Examples**: True/False, Yes/No, Spam/Not Spam, Positive/Negative sentiment.
2. **Multiclass classification**:
   - **Definition**: Involves more than two categories or classes.
   - **Examples**: Classifying animals into Cats/Dogs/Birds, predicting the genre of a book (e.g., Fiction/Non-fiction/Science/History).
3. **Multilabel classification**:
   - **Definition**: Each instance can belong to multiple classes simultaneously.
   - **Examples**: Tagging a research article with multiple topics like "Machine Learning," "Natural Language Processing," and "Healthcare.", Assigning multiple genres to a book.

Simple pipeline



Opening the pipeline



Preparing data

- Data collection
- Text pre-processing
- Data splitting
  - Training
  - Validation (development)
    - to tune the hyperparameters
  - Test
- Text representation

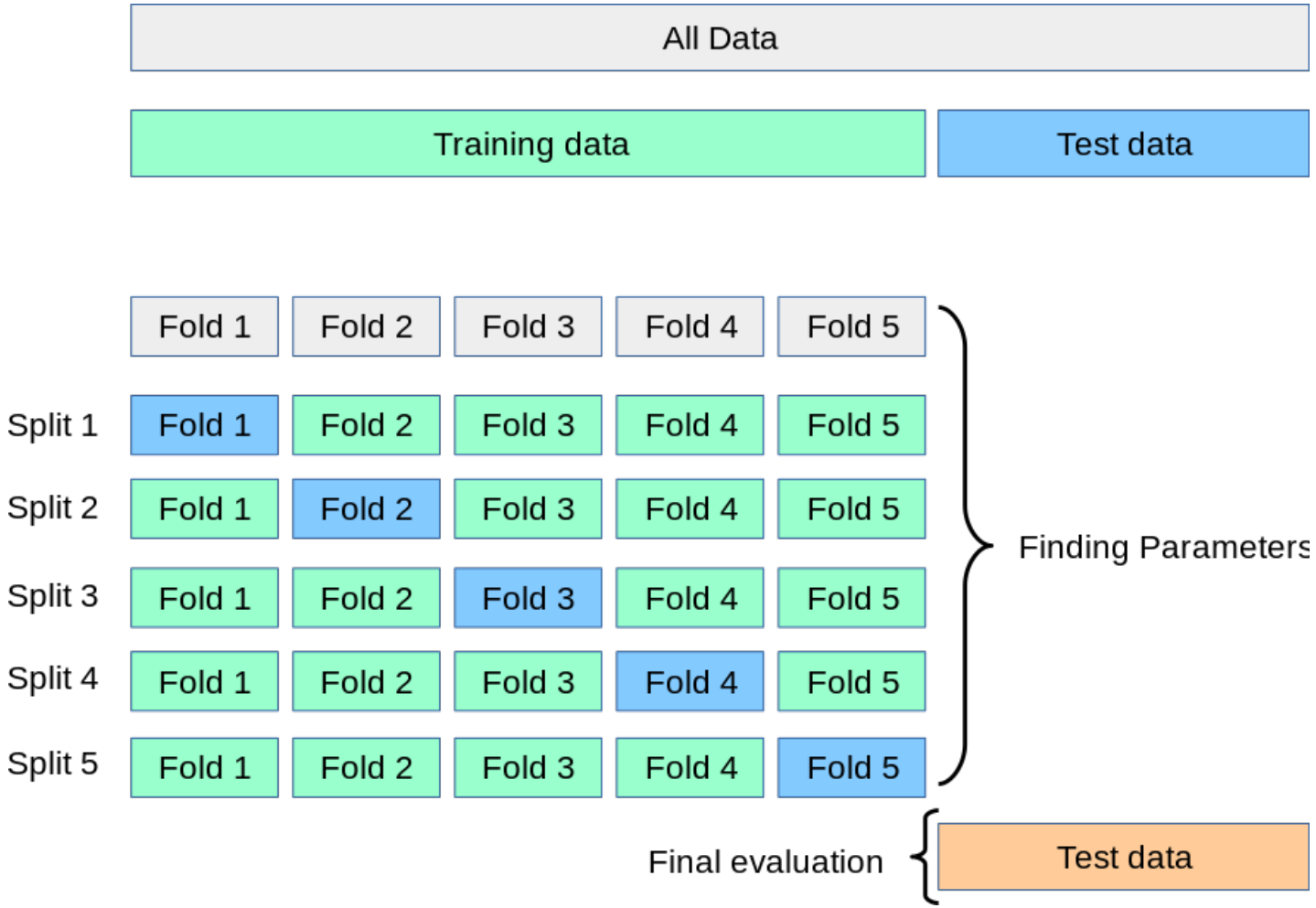
Data splitting

- **Methods:**
  - Train-Test Split
    - Split the dataset into two parts: training and testing (e.g., 80%-20%).
  - K-fold cross validation
    - Split the dataset into K non-overlapping folds of approximately equal size.
  - Leave-One-Out Cross-Validation
    - A special case of K-Fold where K=N (number of data points).

K-fold cross validation

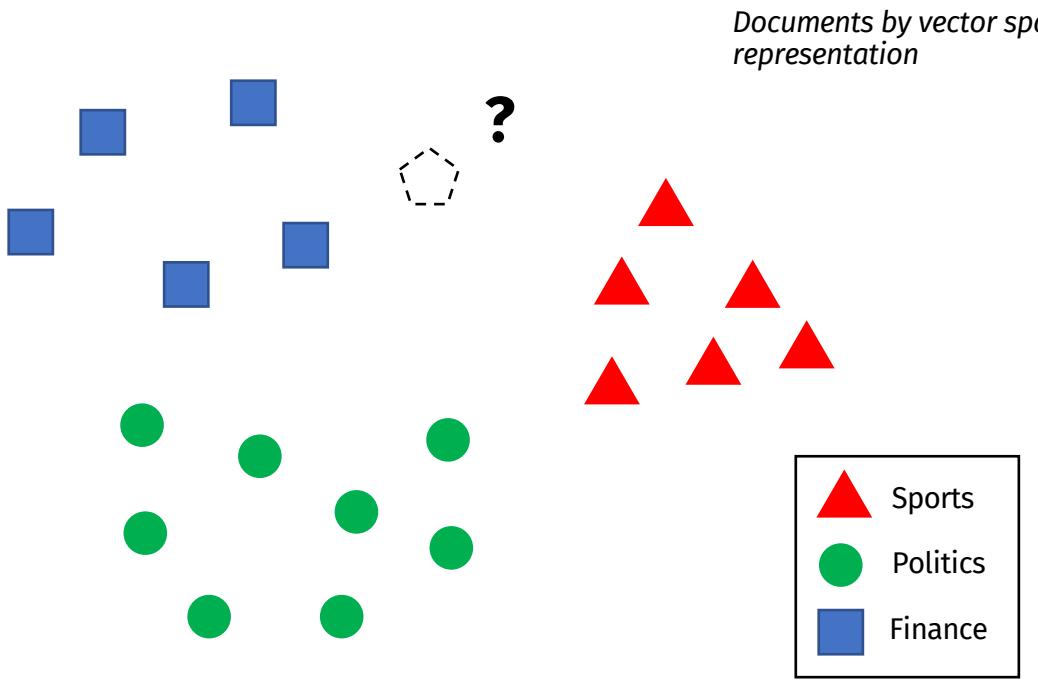
1. **Purpose**:
   - Evaluates the performance of a model on unseen data by splitting the dataset into K subsets (folds).
   - Ensures that every data point gets a chance to be in both the training and validation sets.
2. **How it Works**:
   - Split the data into K equally-sized folds.
   - Train the model on K−1 folds and validate it on the remaining fold.
   - Repeat the process K times, each time using a different fold for validation.
3. **Advantages**:
   - Reduces the risk of overfitting compared to a single train-test split.
   - Provides a more robust estimate of the model's generalization performance.

K-fold cross validation



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

How to classify this document?



Text Classification: definition

- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*: a predicted class $c \in C$

Classification Algorithms

Hand-coded rules

- Rules based on combinations of words or other features
- Accuracy can be high: If rules carefully refined by expert
- But building and maintaining these rules is expensive
- Data/Domain specifics

Supervised Machine Learning

- *Input:*
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$
- *Output:*
  - a learned classifier $y:d \rightarrow c$

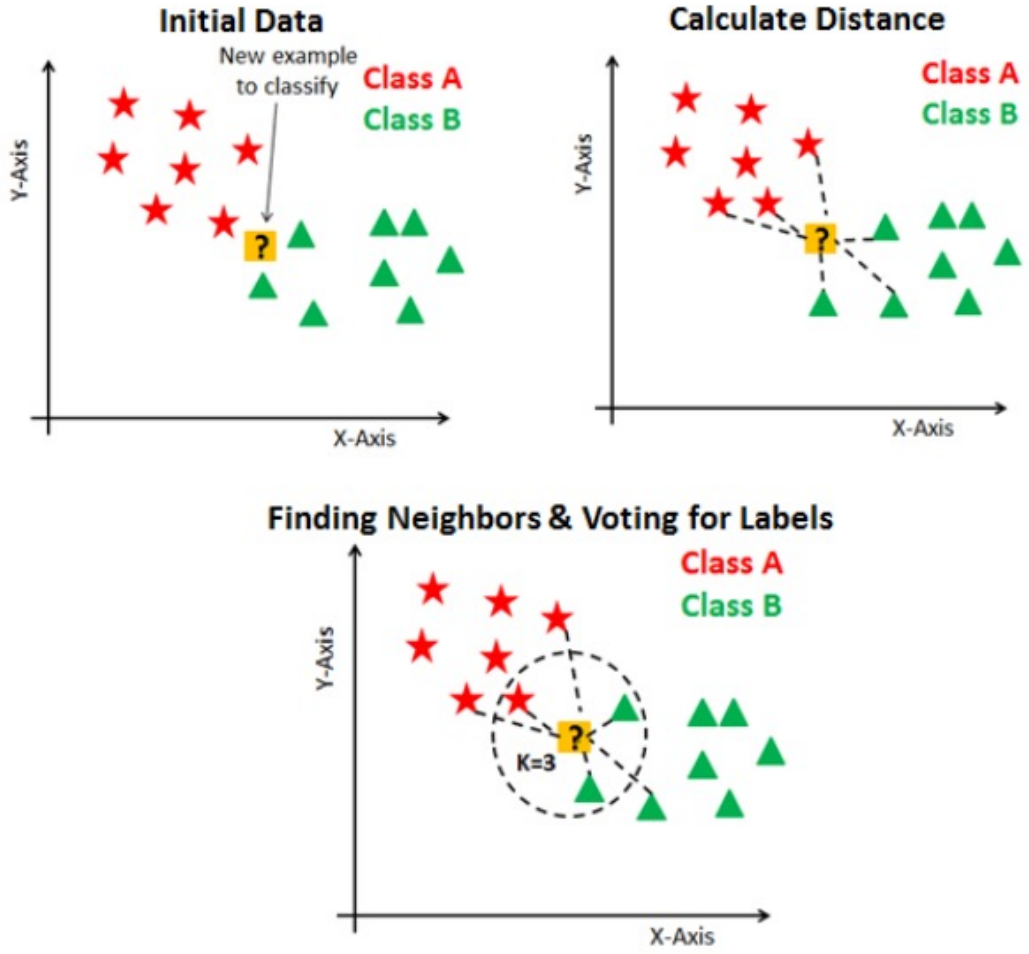
Common algorithms

- K-nearest neighbors
- Naïve bayes
- Logistic regression
- Support vector machines
- Neural networks
- Deep learning

K-nearest neighbors

- **K-Nearest Neighbors (KNN)** is a simple, non-parametric algorithm that classifies data points based on the majority class of their K nearest neighbors in feature space, using a distance metric such as Euclidean distance.

- Non-parametric methods: make no strong assumptions about the data's underlying distribution.



Naive Bayes

- **Naive Bayes** is a probabilistic machine learning algorithm based on Bayes' Theorem, which assumes that all features are independent of each other (a "naive" assumption) and uses this simplification to classify data efficiently.

y(

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

)=c

Bayes' rule

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Learning naive Bayes

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}}\, \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}}\, P(d \mid c)P(c)$$

Dropping the denominator

Learning naive Bayes

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

Document d represented as features x1..xn

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Could only be estimated if a very, very large number of training examples was available.

We can just count the relative frequencies in a corpus

Learning naive Bayes

- Simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

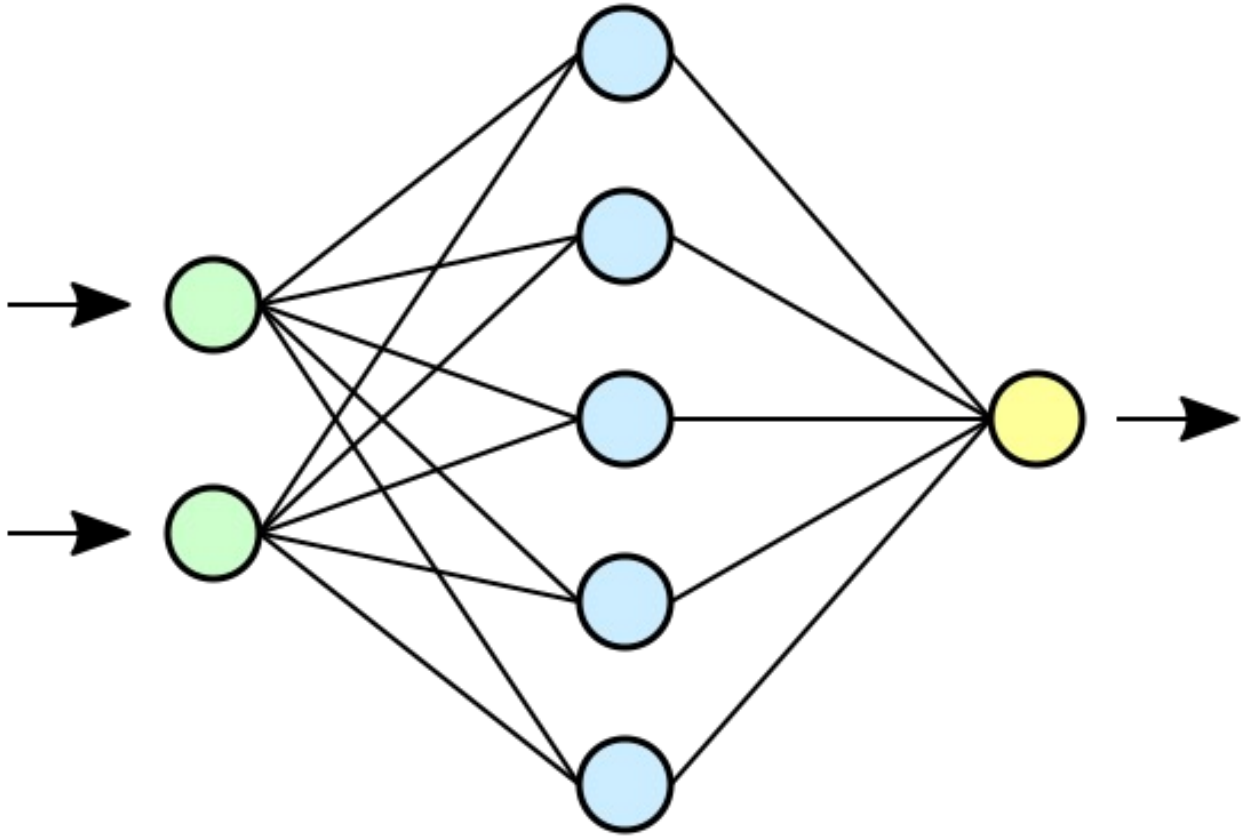
$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

Neural networks, “deep learning”

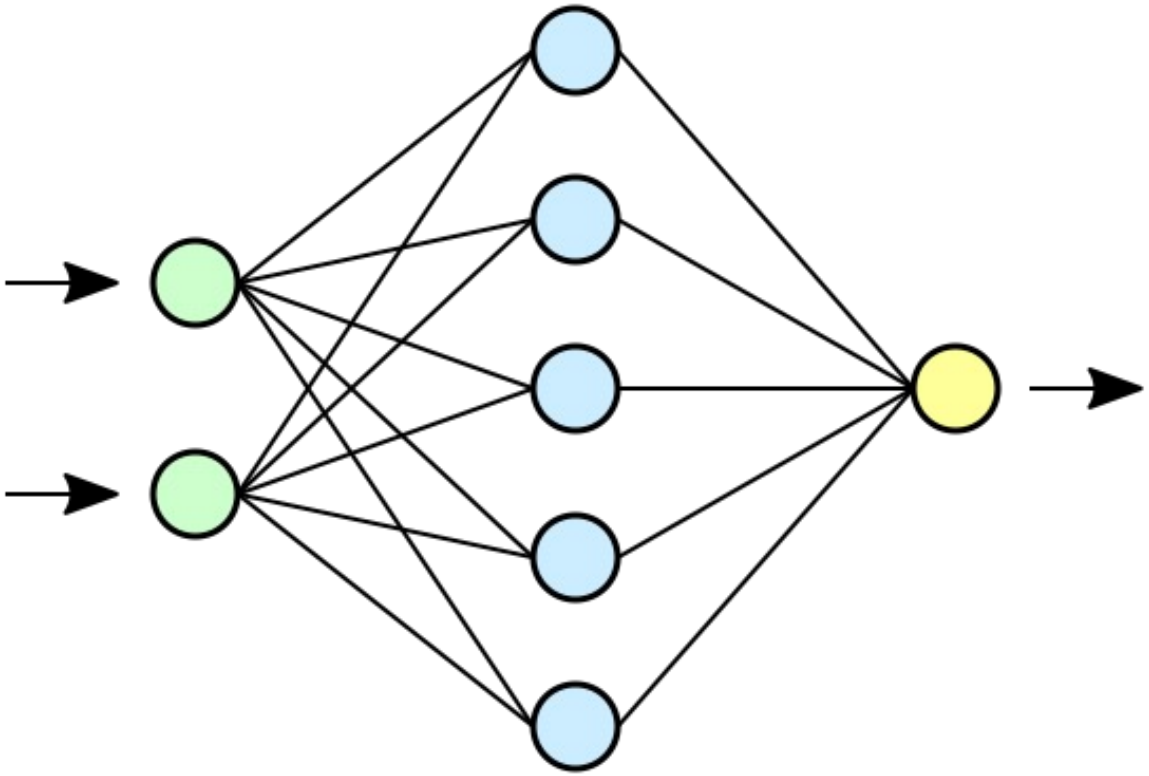
- **Neural networks** are machine learning models inspired by the structure of the human brain, consisting of layers of interconnected nodes (neurons) that process and learn patterns in data by adjusting weights through a process called backpropagation.
  - Compositional approach to curve-fitting;
  - “Biologically inspired”

    (but don’t take that too seriously);
  - Sound cool.

Neural network



Neural network



"Hidden" nodes:

Example:

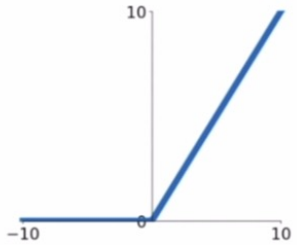
$$h_1 = f(w_{11}x_1 + w_{12}x_2)$$

Output:

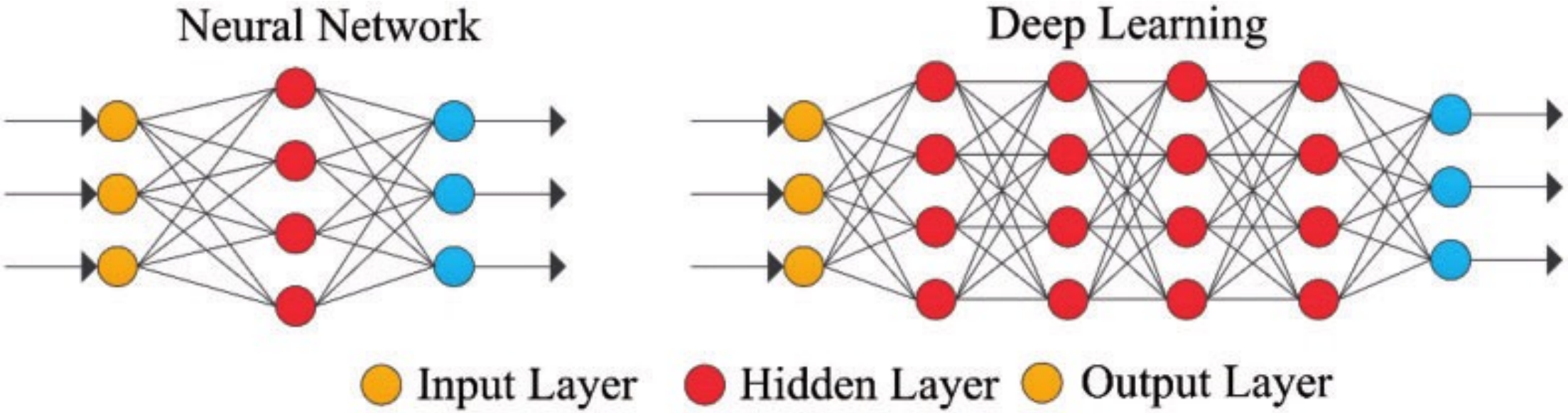
$$y = f(w_{21}h_1 + w_{22}h_2 + \cdots + w_{25}h_5)$$

"Activation function":

- RElu $f(z) =$
- …



What makes a neural net “deep”?



More on neural networks

- **Weeks 6, 7**

Ensemble classifiers

- We can combine multiple classifiers
  - **Random forest classifier**
    - fit multiple decision trees, average over predictions
    - Less “volatile” than a single decision tree
  - **Voting classifier**
    - Fit multiple classifiers and let them “vote” on the result
    - Classifiers may be of different classes!

Selecting hyperparameters

- A hyperparameter is a parameter whose value is used to control the learning process, which must be configured before the process starts.
  - Experts input
  - Hyperparameter optimization: the selection process will be automated
    - Random search
    - Grid search: systematically go through your data
    - Other optimization techniques

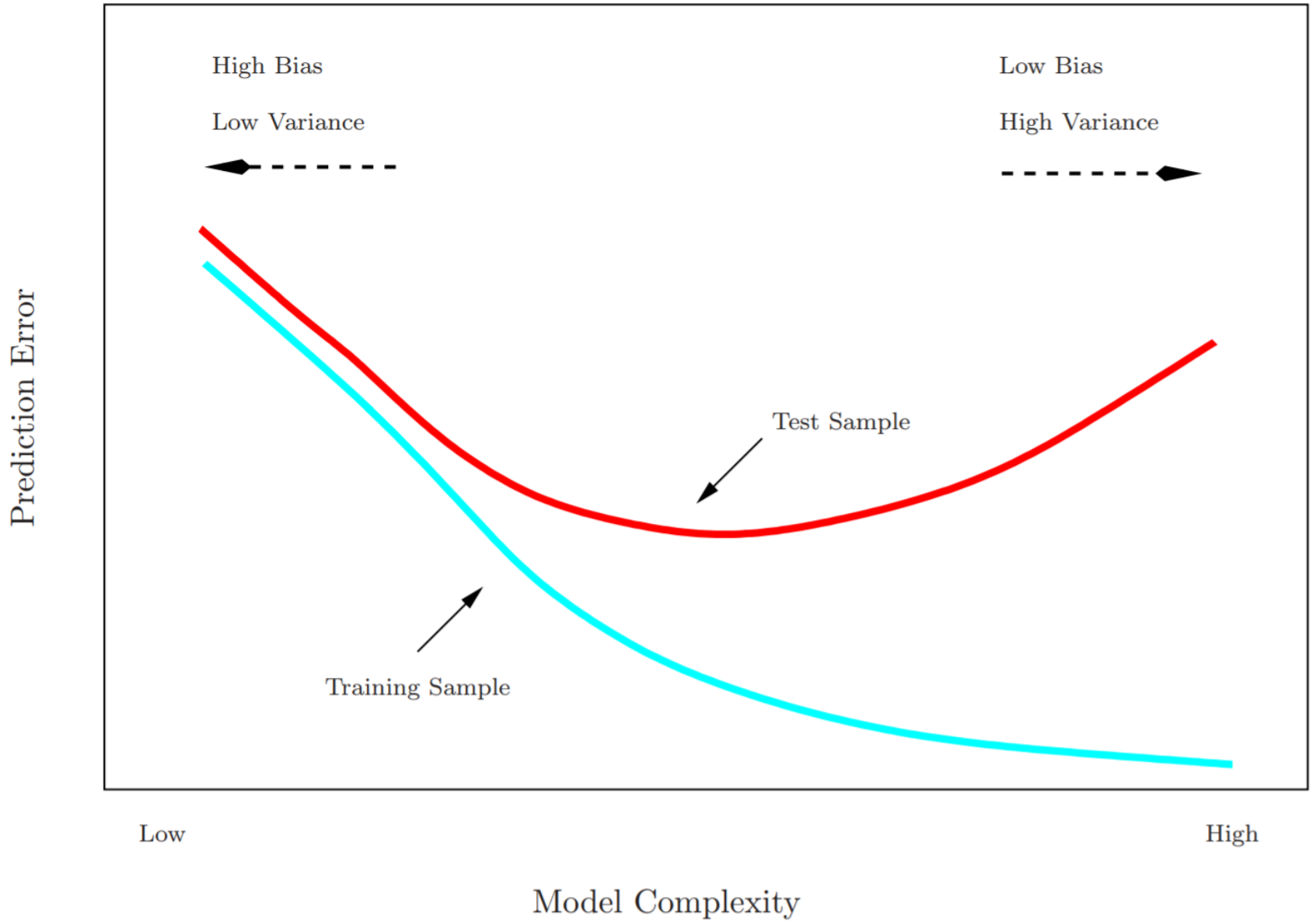
Break

Evaluation

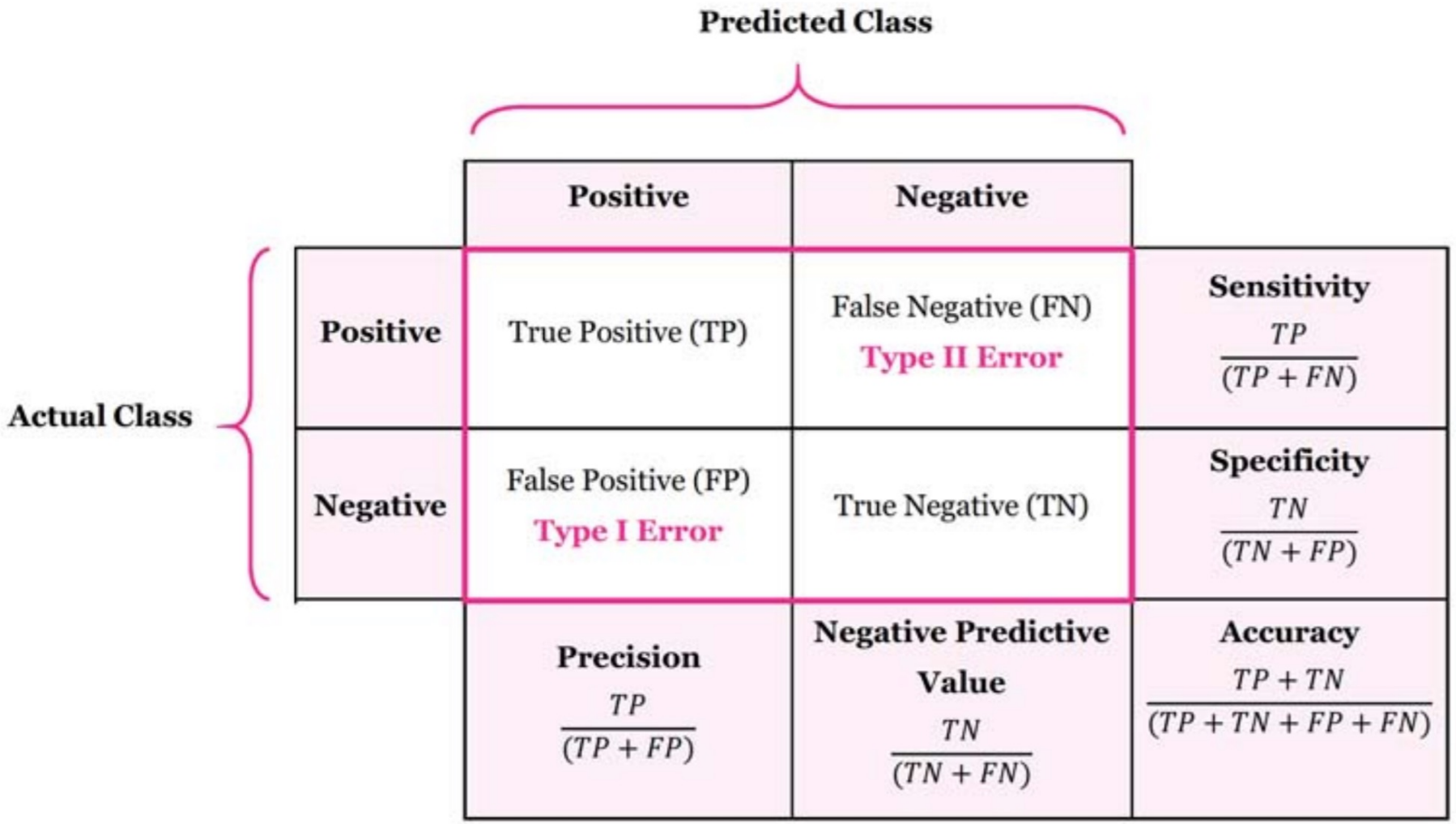
No free lunch

“Any two optimization algorithms are equivalent when their performance is averaged across all possible problems”

(Wolpert & MacReady)


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

Confusion matrix



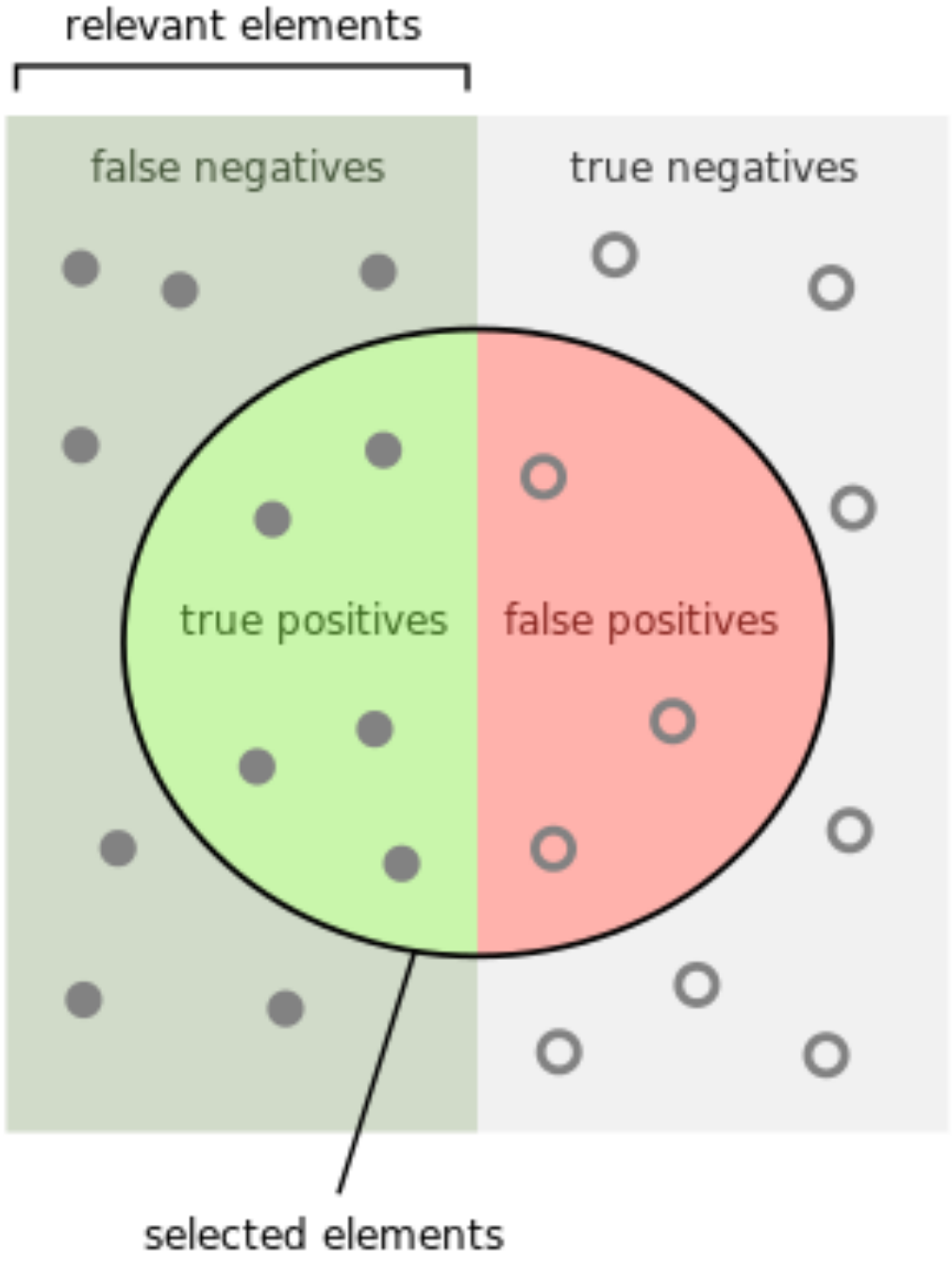
Accuracy

- Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed.

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

- Precision is a valid choice of evaluation metric when we want to be very sure of our prediction.
- Recall is a valid choice of evaluation metric when we want to capture as many positives as possible.



How many selected items are relevant?

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

How many relevant items are selected?

$$\text{Recall} = \frac{\text{green}}{\text{green + olive}}$$

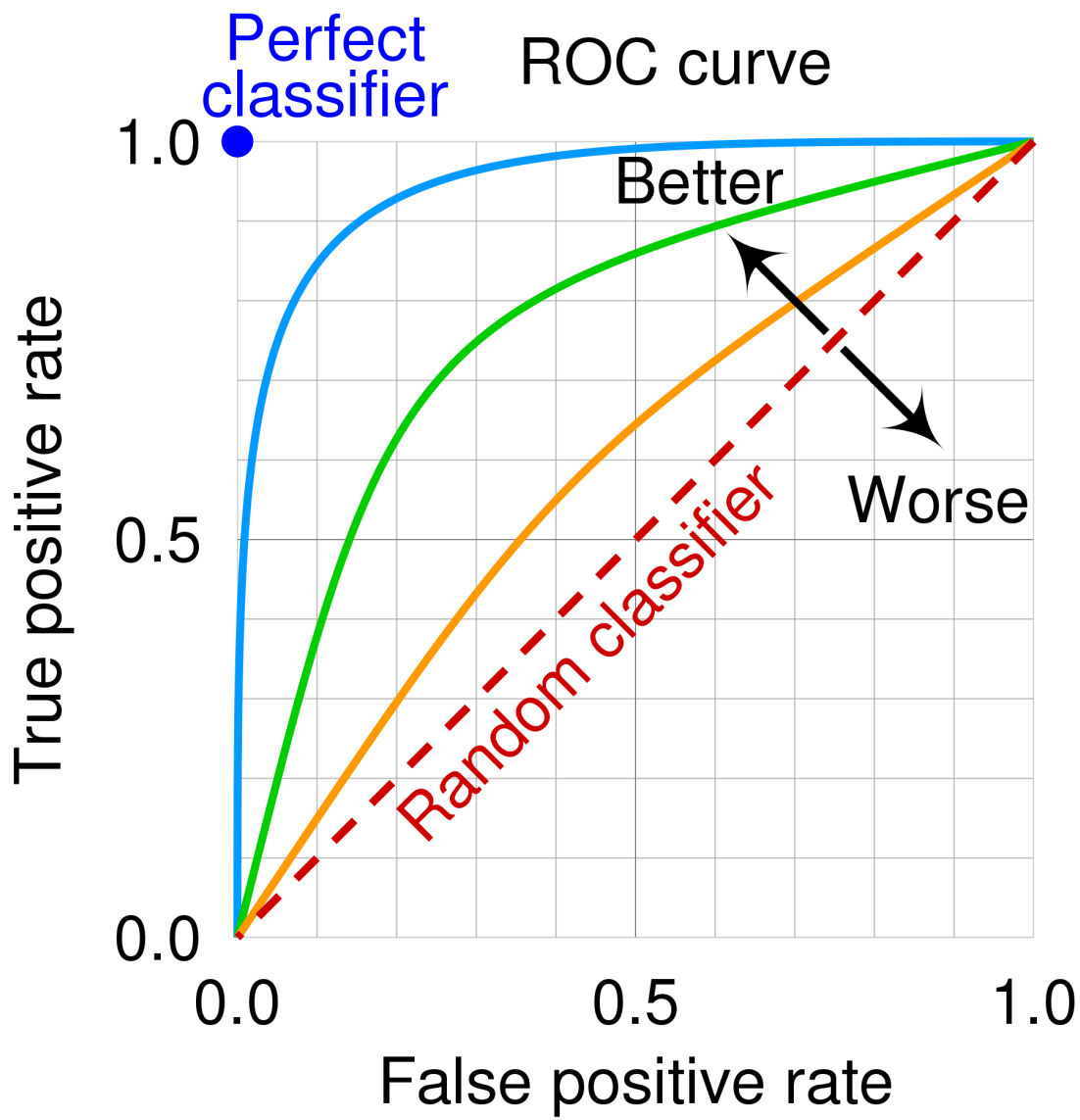
Source: https://en.wikipedia.org/wiki/F-score

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- The harmonic mean is a very conservative average
- People usually use balanced F1 measure
  - i.e., with $\beta = 1$ (that is, $\alpha = ½$): $F = 2PR/(P+R)$


Perfect classifier
ROC curve
1.0
Better
0.5
Worse
Random classifier
True positive rate
0.0
0.0
0.5
1.0
False positive rate

Evaluation metrics: multiclass classification

- Evaluation is less straightforward with multiple classes
- Precision/recall/F1 per class
- But what if we want a single score?
  - **Macro F1:** average of F1 score for each class
  - **Micro F1:** uses total number of true/false positives/negatives

Conclusion

- Understanding the classification pipeline is essential for building effective models, from preprocessing text data to training and evaluation.
- There are different algorithms available for text classification, each with its strengths.
- Accurate evaluation using metrics like accuracy, precision, recall, and cross-validation ensures reliable model performance and generalization to unseen data.

Practical 3: Multiclass text classification of 20newsgroups articles.

Questions?