UNIVERSITÉ DU LUXEMBOURG

Natural Language Processing

Lecture 5: Text Classification (Part 1)

Salima Lamsiyah
University Luxembourg, FSTM, DCS, MINE Research Group
Salima.lamsiyah@uni.lu

Recap!

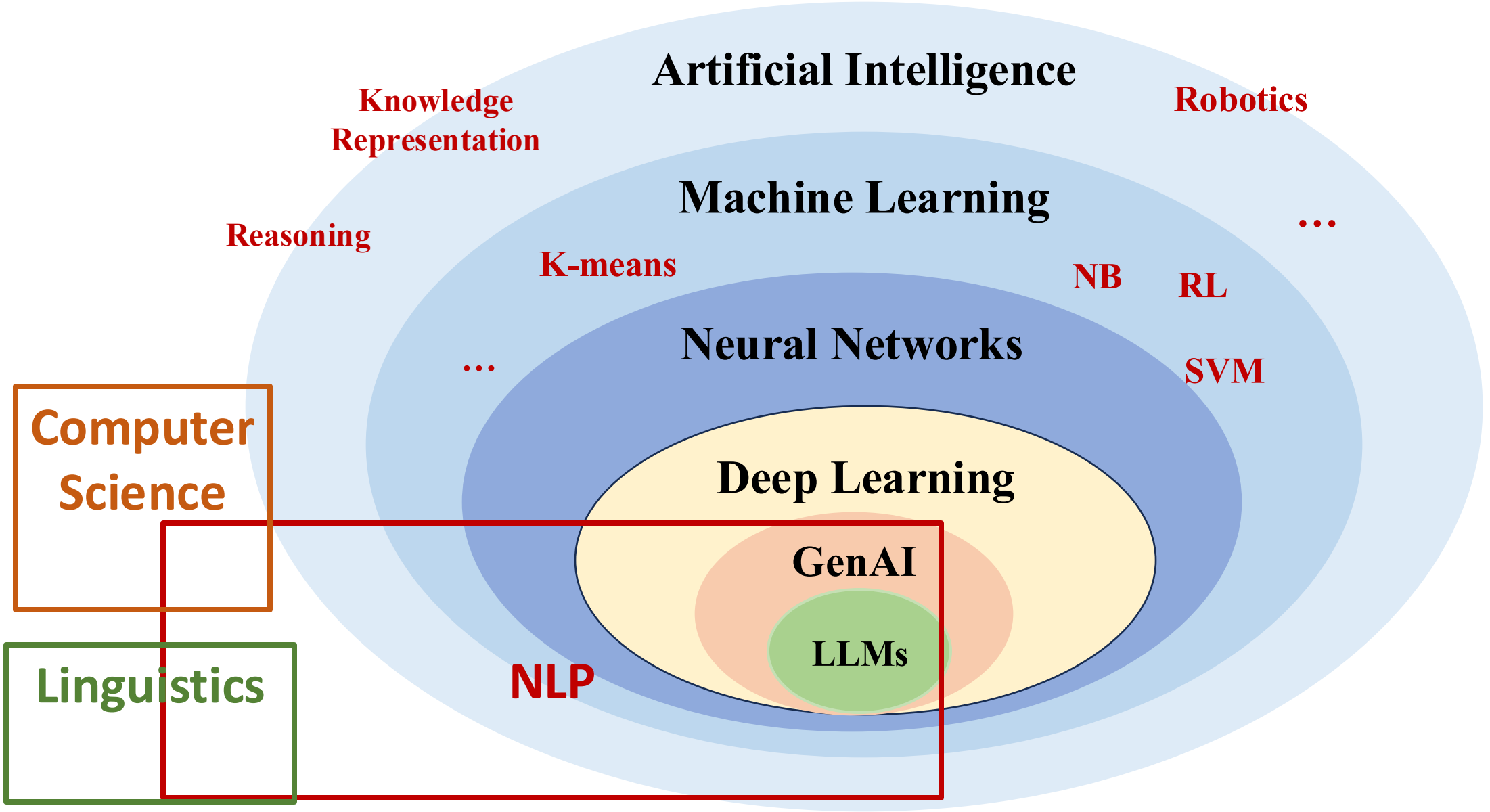
1. Introduction to NLP
2. Text Preprocessing Techniques
3. Text Representations Methods
   - Bag-of-words (One-hot encoding, TF, TF-IDF, PMI)
   - Word Embedding
   - Contextual Embedding
4. Language Modeling (N-grams, Laplace Smooting, Perplexity, backoff, Interpolation)

Lecture Plan

1. Quick Review on Supervised Learning and Probabilities
2. Text Classification
3. Generative Model: Naïve Bayes
4. Exercises
5. How to evaluate a Classifier?

Quick Review

AI Terminology: Artificial Intelligence



Machine Learning: *Definition*

- [Authur Samuel 1959]: Machine learning is a subfield of Artificial Intelligence that gives Computers the Ability to Learn from Data, without being explicitly programmed.

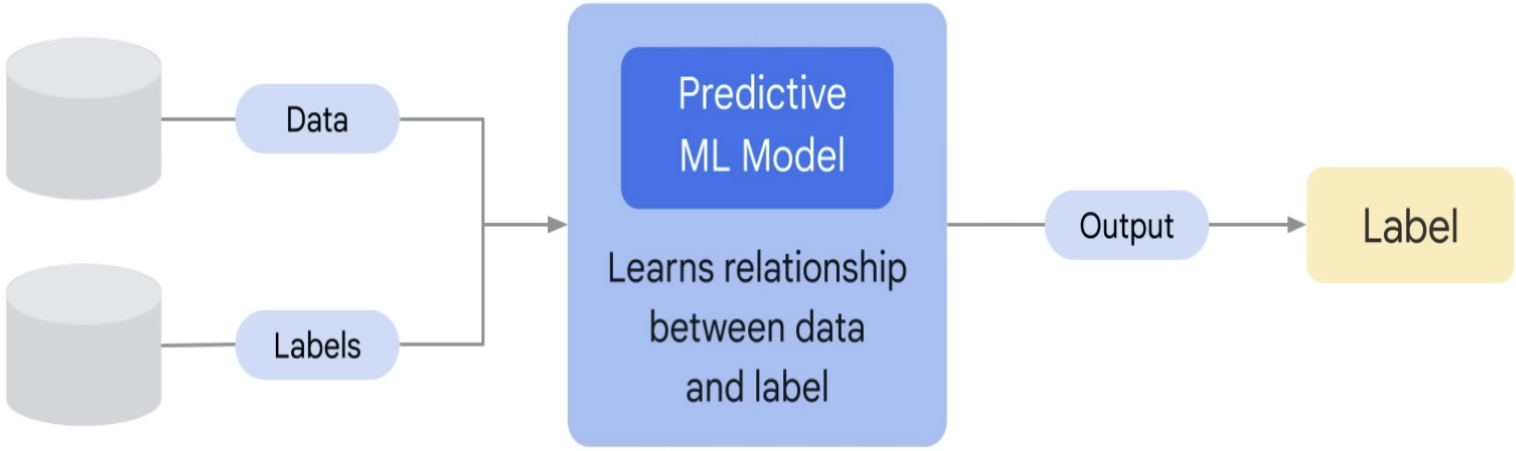
- Machine Learning algorithms construct predictive models by learning from a large number of training examples.

- By using machine learning, the computer can learn to recognize patterns and make decisions based on data, without requiring explicit programming instructions for every possible scenario.

Machine Learning: *Learning Paradigm*

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Reinforcement Learning
4. Semi-Supervised Machine Learning
5. Self-Supervised Machine Learning

Supervised Machine Learning algorithms

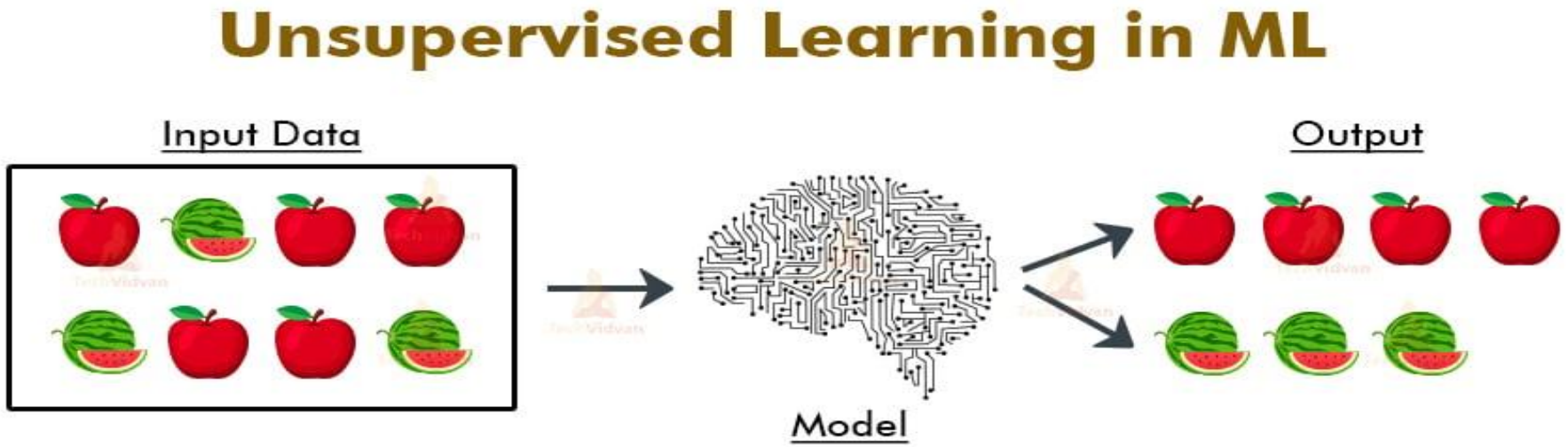


Learns from being given ***"right answers"***

Source

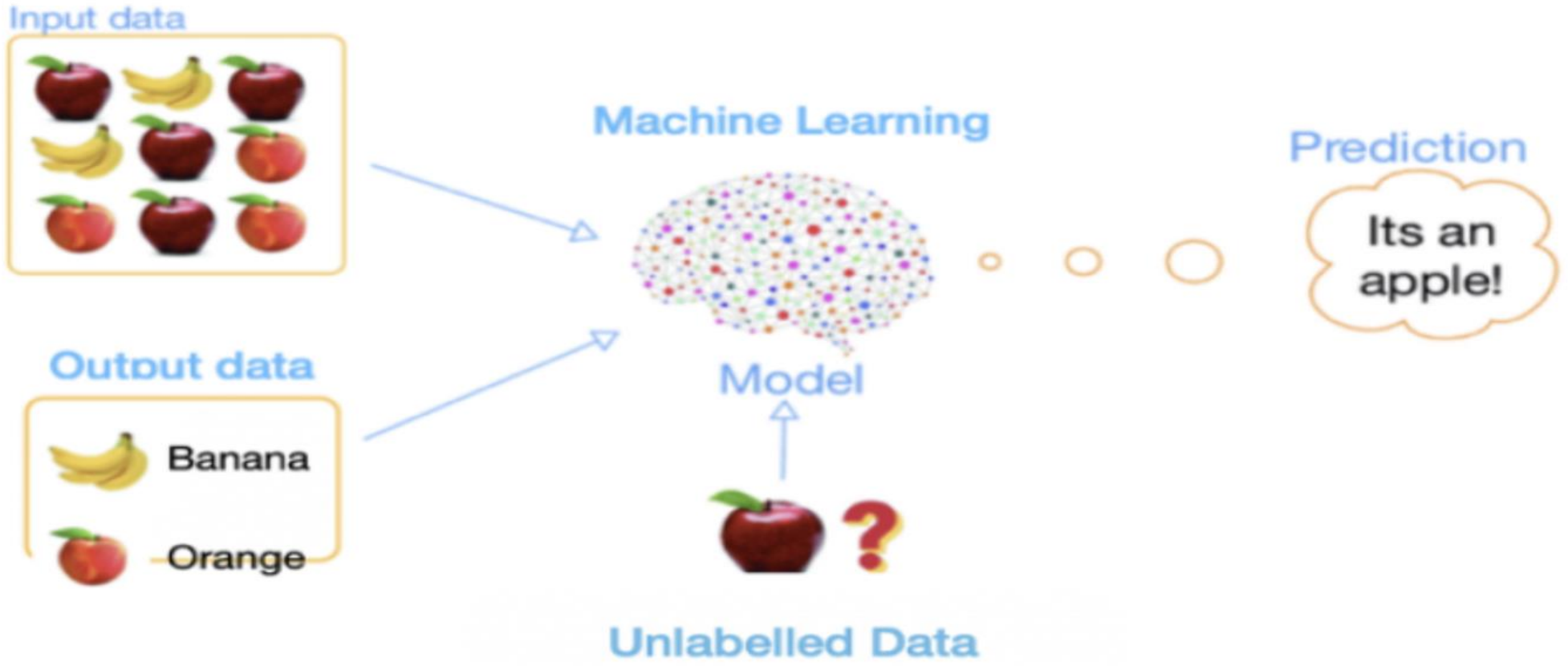
- **Examples of Supervised Learning Algorithms include:**
- Linear Regression, Logistic Regression Support Vector Machines, Decision Trees, Naïve Bayes,
- Perceptrons
- Deep Learning models (e.g., Feed-Forward NNs, Recurrent NNs, Convolutional NNs)

Unsupervised Machine Learning



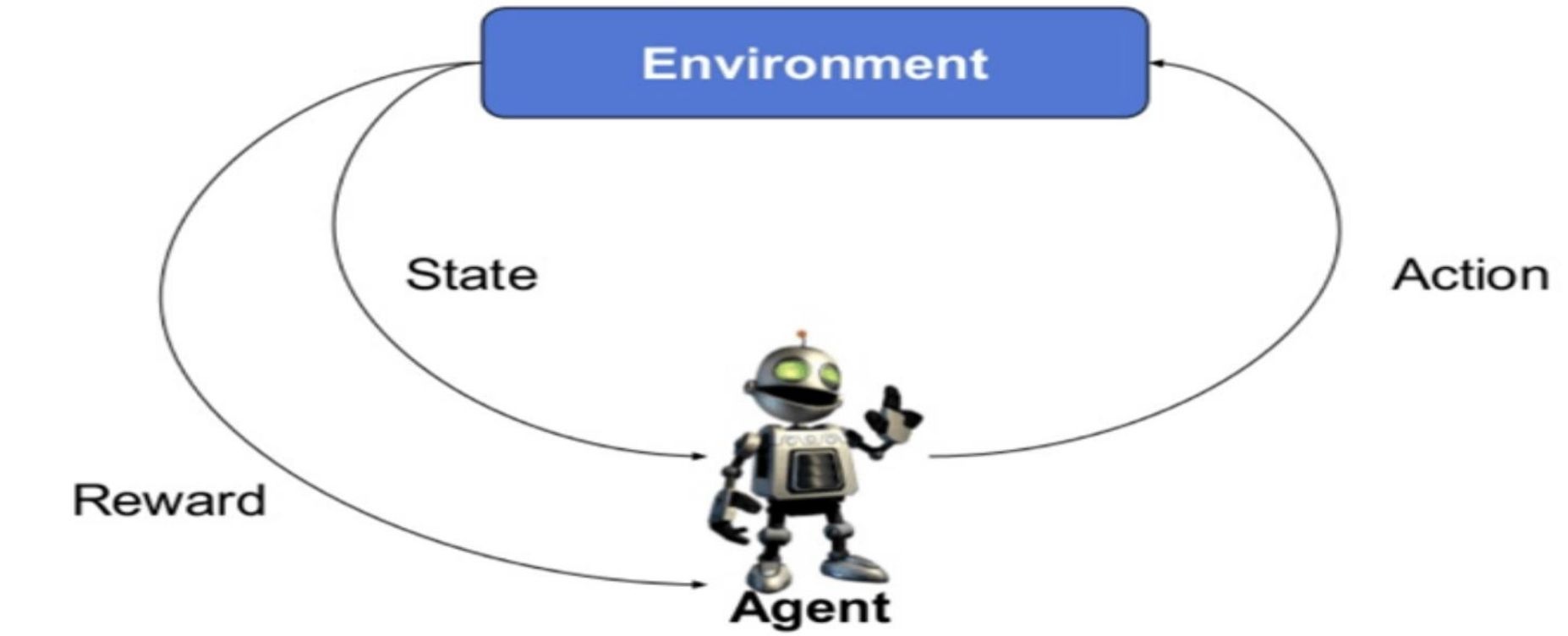
- **Examples of Unsupervised Learning Algorithms include:**
  - Clustering algorithms (e.g., k-means clustering, hierarchical clustering),
  - Deep Learning models (e.g., autoencoders, variational autoencoders, generative adversarial networks).

Semi-Supervised Machine Learning



Source

Reinforcement Learning



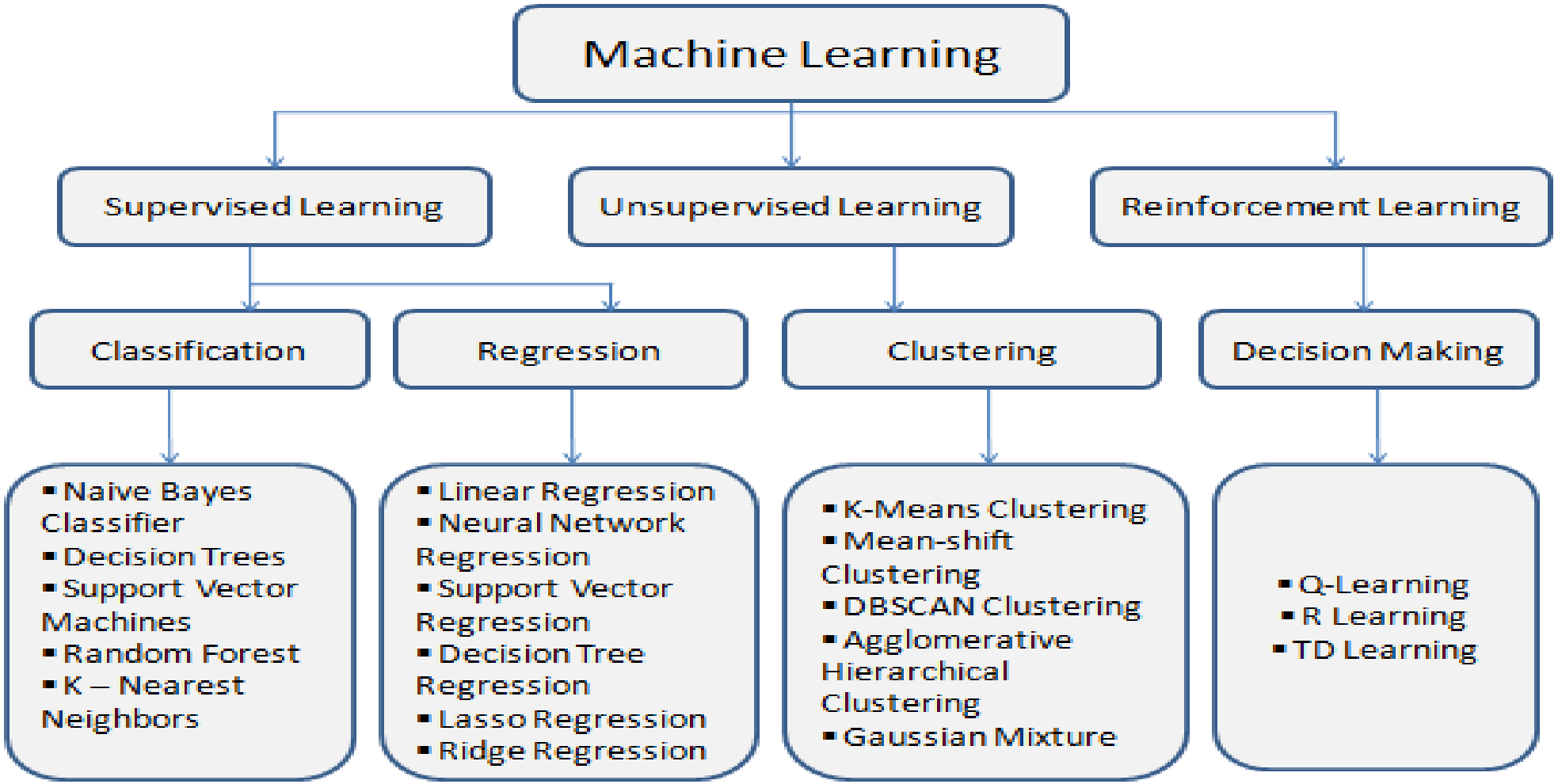
- **Examples of RL applications include:**
  - Game playing: Chess, Go, Atari, Tic-Tac-Toes games
  - Robotics, Autonomous vehicles, …

Self-Supervised Machine Learning

- In Self-Supervised Machine Learning unstructured data is provided as input to the model.
  - The model trains itself to learn one part of the input from another part of the input
  - The model annotates the data on its own, and labels that have been predicted with high confidence are used as ground truths in future iterations of the model training
  - Examples: BERT, GPT-5, Gemini, …

- Check link

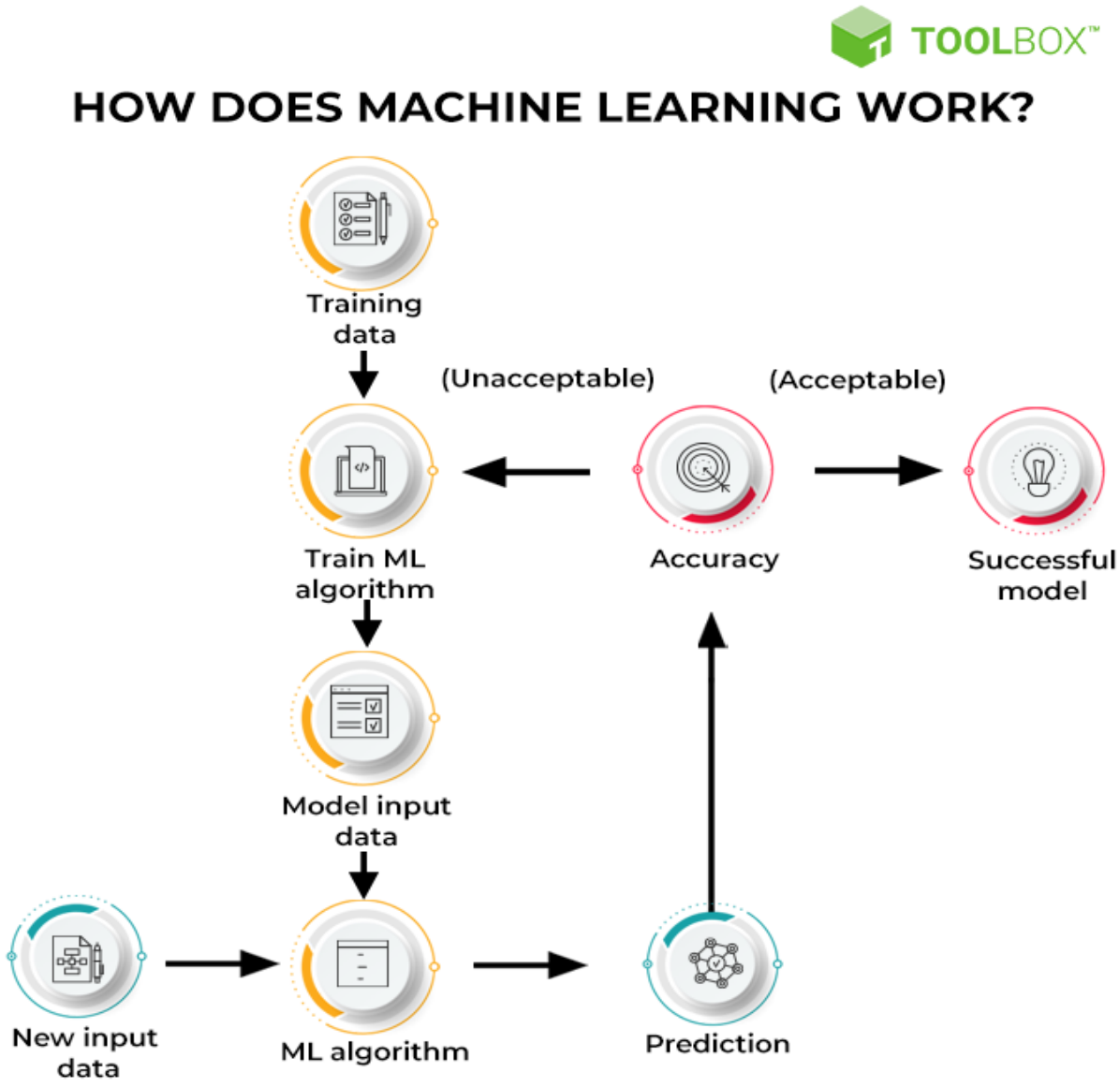
Types of Machine Learning Models



Source

To Summarize

- Machine learning gives Computers the Ability to Learn from Data.
- Machine Learning allows building intelligent systems to transform data into knowledge.
- Machine Learning algorithms construct predictive models by learning from a large number of training examples.



Source

Supervised Learning Output: Numerical or Categorical Values (This class)

- **Regression:**

The output variable to be predicted is continuous in nature, e.g. scores of a student, diamond prices, house prices, etc.

  - Linear Regression Algorithm
  - Polynomial Regression Algorithm
  - …

- **Classification:**

The output variable to be predicted is categorical in nature, e.g. classifying incoming emails as spam or ham, Yes or No, True or False, 0 or 1.

  - Naïve Bayes
  - Logistic Regression Algorithm
  - SVM
  - K-NN
  - ..

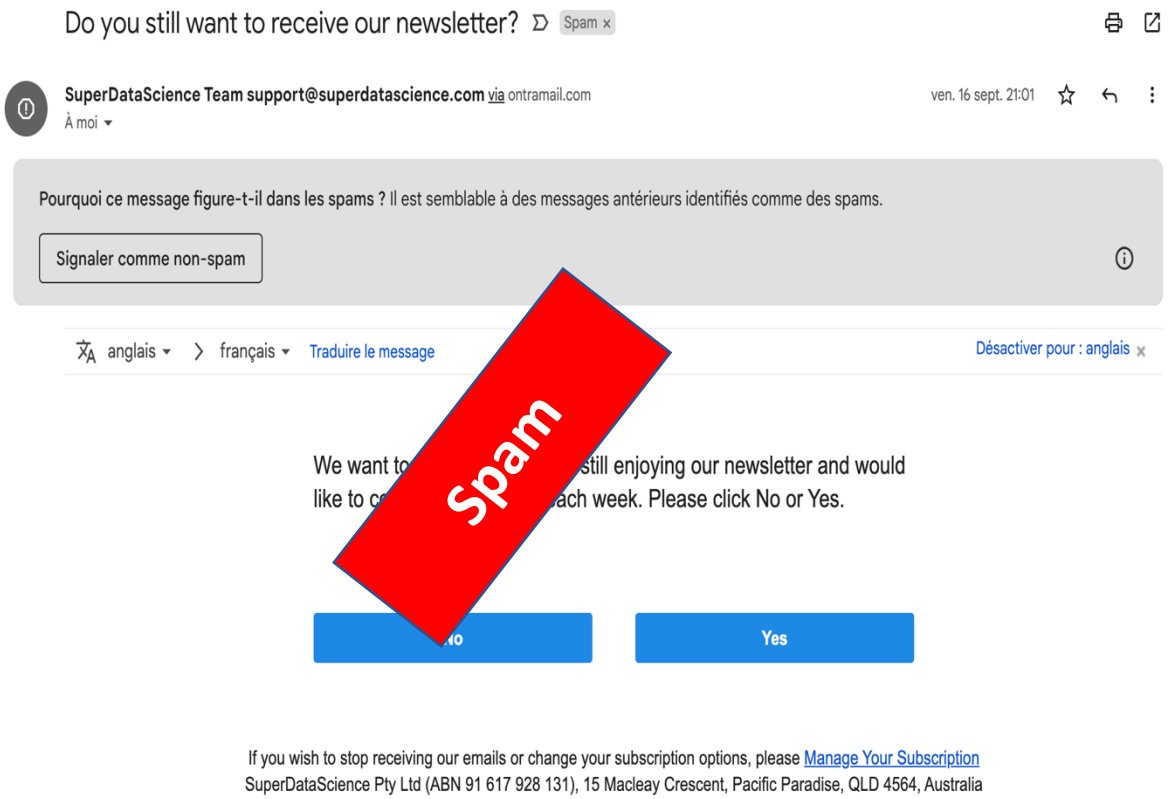
This Class (Supervised Machine Learning)
Text Classification

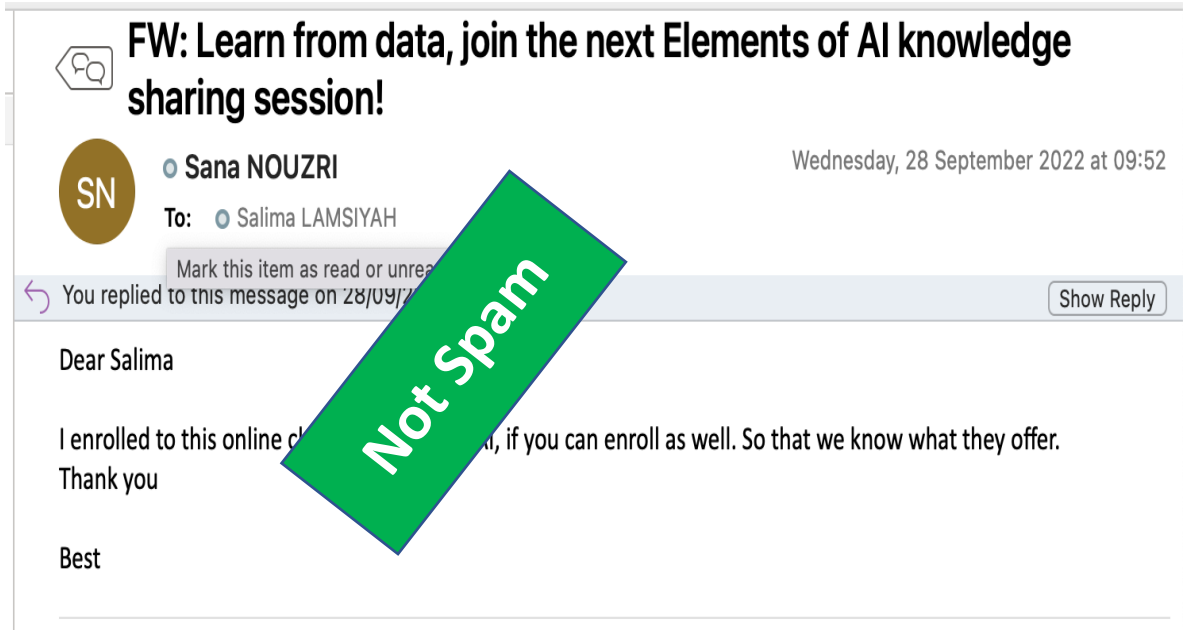
1. Collect Data
2. Text Preprocessing
   - Tokenization
   - Stemming
   - POS …
3. Text Representation (BOW, TF-IDF, PMI)
4. Machine Learning Model Selection and Training
5. Evaluation of the Selected Model

- Many kinds of classifiers!
  - Logistic regression
  - Naive Bayes
  - Neural networks
  - *k*-nearest neighbors
  - …
  - LLMs
    - Fine-tuned as classifiers
    - Prompted to give a classification

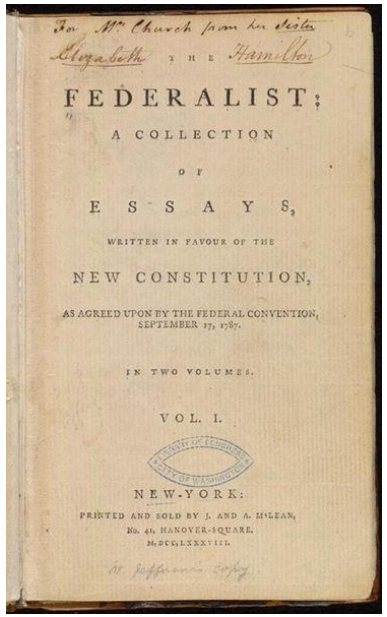
Text Classification

Is this spam?

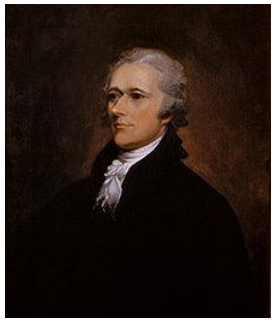

Do you still want to receive our newsletter?
Spam
SuperDataScience Team support@superdatascience.com via ontramail.com
ven. 16 sept. 21:01
À moi
Pourquoi ce message figure-t-il dans les spams ? Il est semblable à des messages antérieurs identifiés comme des spams.
Signaler comme non-spam
anglais
français
Traduire le message
Désactiver pour : anglais
Spam
We want to
still enjoying our newsletter and would
like to c
ach week. Please click No or Yes.
Yes
If you wish to stop receiving our emails or change your subscription options, please Manage Your Subscription
SuperDataScience Pty Ltd (ABN 91 617 928 131), 15 Macleay Crescent, Pacific Paradise, QLD 4564, Australia


FW: Learn from data, join the next Elements of AI knowledge sharing session!
SN
Sana NOUZRI
Wednesday, 28 September 2022 at 09:52
To: Salima LAMSIYAH
Mark this item as read or unrea
You replied to this message on 28/09/
Show Reply
Dear Salima
I enrolled to this online c
, if you can enroll as well. So that we know what they offer.
Thank you
Best
Not Spam

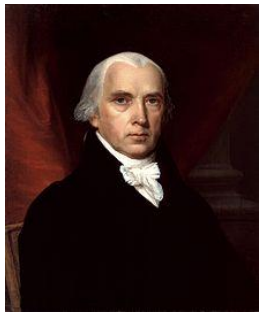
Who wrote which *Federalist Papers*?



- 1787-8: essays anonymously written by:
- Alexander Hamilton, James Madison, and John Jay
- to convince New York to ratify U.S Constitution
- Authorship of 12 of the letters unclear between:



Alexander Hamilton



James Madison

- 1963: solved by Mosteller and Wallace using Bayesian methods

Wikimedia commons

Topic Categorization

- What is the topic of this NLP article?

Adversarial Multi-task Model for Emotion, Sentiment, and Sarcasm Aided Complaint Detection

Apoorva Singh[1], Arousha Nazir[2], and Sriparna Saha[1](✉)

[1] Indian Institute of Technology Patna, Bihta, India
{apoorva_1921cs19,sriparna}@iitp.ac.in
[2] National Institute of Technology Srinagar, Srinagar, India

Abstract. Automatic identification of consumer complaints about products or services purchased can be crucial for businesses and online merchants since they can utilize this knowledge to address the needs of their clients, including handling and resolving complaints. Previous studies on complaint detection do not consider sarcasm, which is often used to express a breach of expectation without directly stating the complaint. Furthermore, since every speech act is influenced by emotions, the customer's emotional state has a considerable impact on the complaint expression. In this paper, we hypothesize that sarcasm, along with two closely related tasks of sentiment and emotion, could aid the process of complaint identification and thereby propose a deep multi-task framework to solve the four problems jointly. We manually annotate the recently released *Complaints* dataset with the emotion, sentiment, and sarcasm classes. We present an attention-based adversarial multi-task deep neural network model for complaint detection. Experimental results on the extended version of the *Complaints* dataset show the effectiveness of our proposed approach for complaint detection over the existing state-of-the-art system. The evaluation also demonstrates that the proposed multi-task system improves performance for the primary task, i.e., complaint detection, with the assistance of the three auxiliary tasks, emotion recognition, sentiment analysis, and sarcasm detection.

Keywords: Complaint Detection · Emotion Recognition · Sentiment Analysis · Sarcasm Detection · Adversarial Multi-task learning · Deep learning

1 Introduction

Complaining is a speech act that generally conveys unpleasant emotions that are caused by a disparity between reality and expectations regarding an entity or event [20]. Complaints are fundamental ways of expressing displeasure in human communication. The expression of complaints varies from person to person depending on the complainers' temperament and specific situations [35]. Presently, social

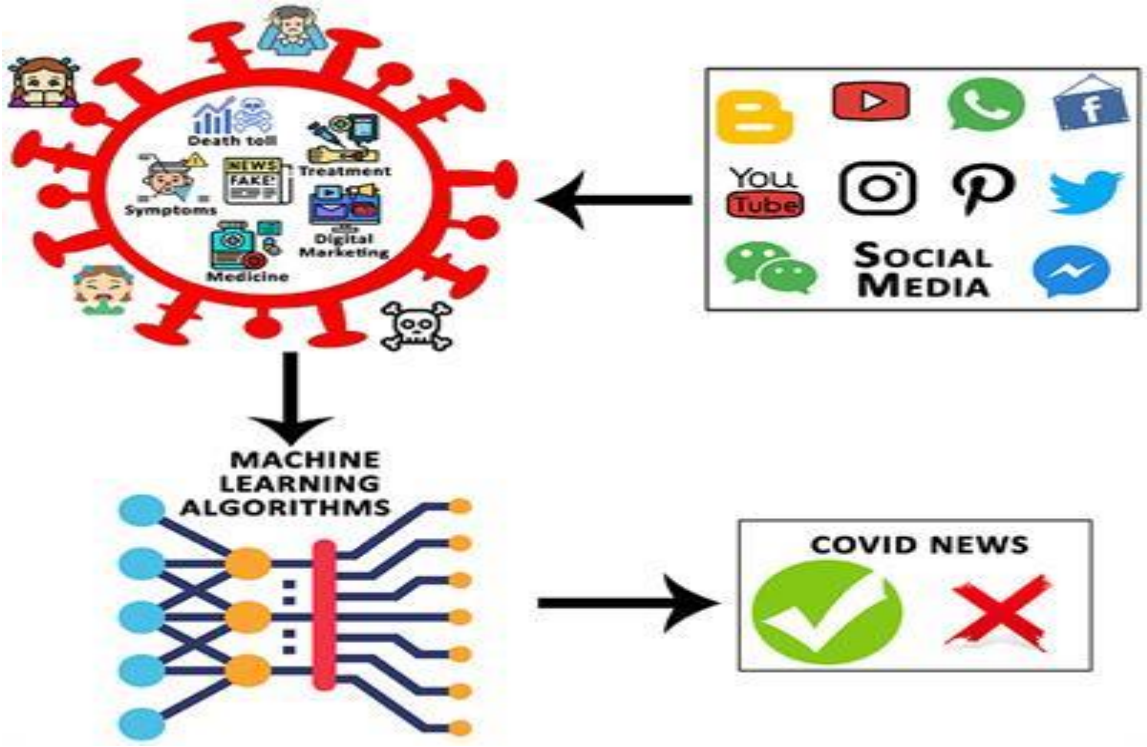
M. Hagen et al. (Eds.): ECIR 2022, LNCS 13185, pp. 428–442, 2022.
https://doi.org/10.1007/978-3-030-99736-6_29

- Sentiment Analysis
- Text Summarization
- Chatbot
- Information Retrieval
- …

The categories should be predefined.

Fact verification: trustworthy or fake?

Have you
Covid19?
Drink Alcohol!



Sentiment Analysis: Movies Rating

positive

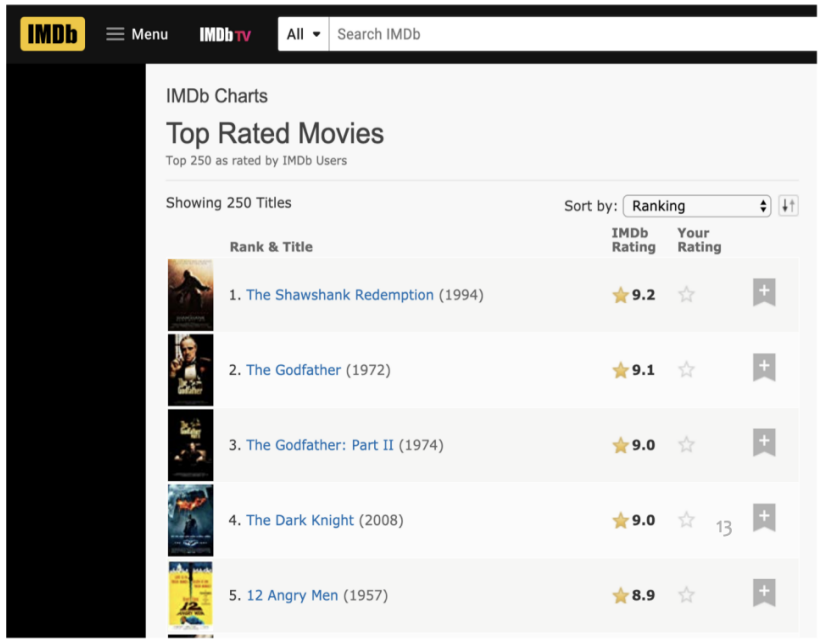
"… is a film which still causes real, not figurative, chills to run along my spine, and it is certainly the bravest and most ambitious fruit of Coppola's genius"

Roger Ebert, Apocalypse Now

- "I hated this movie. Hated hated hated hated hated this movie. Hated it. Hated every simpering stupid vacant audience-insulting moment of it. Hated the sensibility that thought anyone would like it."

  Roger Ebert, North

negative



Sentiment Analysis: Customer Review

By John Neal

This review is from: Accoutrements Horse Head Mask (Toy)

When I turned State's Witness, they didn't have enough money to put me in the Witness Protection Program, so they bought me this mask and some other suggested places to move. Since then I've lived my life in peace and safety knowing that my old identity is forever obscured by this life-saving item.

By Christine E. Torok

Verified Purchase (What's this?)

First of all, for taste I would rate these a 5. So good. Soft, true-to-taste fruit flavors like the sugar variety...I was a happy camper.

BUT (or should I say BUTT), not long after eating about 20 of these all hell broke loose. I had a gastrointestinal experience like nothing I've ever imagined. Cramps, sweating, bloating beyond my worst nightmare. I've had food poisoning from some bad shellfish and that was almost like a skip in the park compared to what was going on inside

Even language modeling can be viewed as classification!

- Let the set of classes be the words (vocabulary V)
- Predicting the next word is classifying
  - the context-so-far
  - into a class for each possible next word

So On!!!

- Spam detection (binary classification: spam/not spam)
- Sentiment analysis (binary or multiway)
- Movie, restaurant, product reviews (pos/neg, or 1-5 stars)
- Political argument (pro/con, or pro/con/neutral)
- Topic classification (multiway: sport/finance/travel/etc)
- Fake News Detection
- Language Identification (multiway: languages, language families)
- …

General Text Classification Definition

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

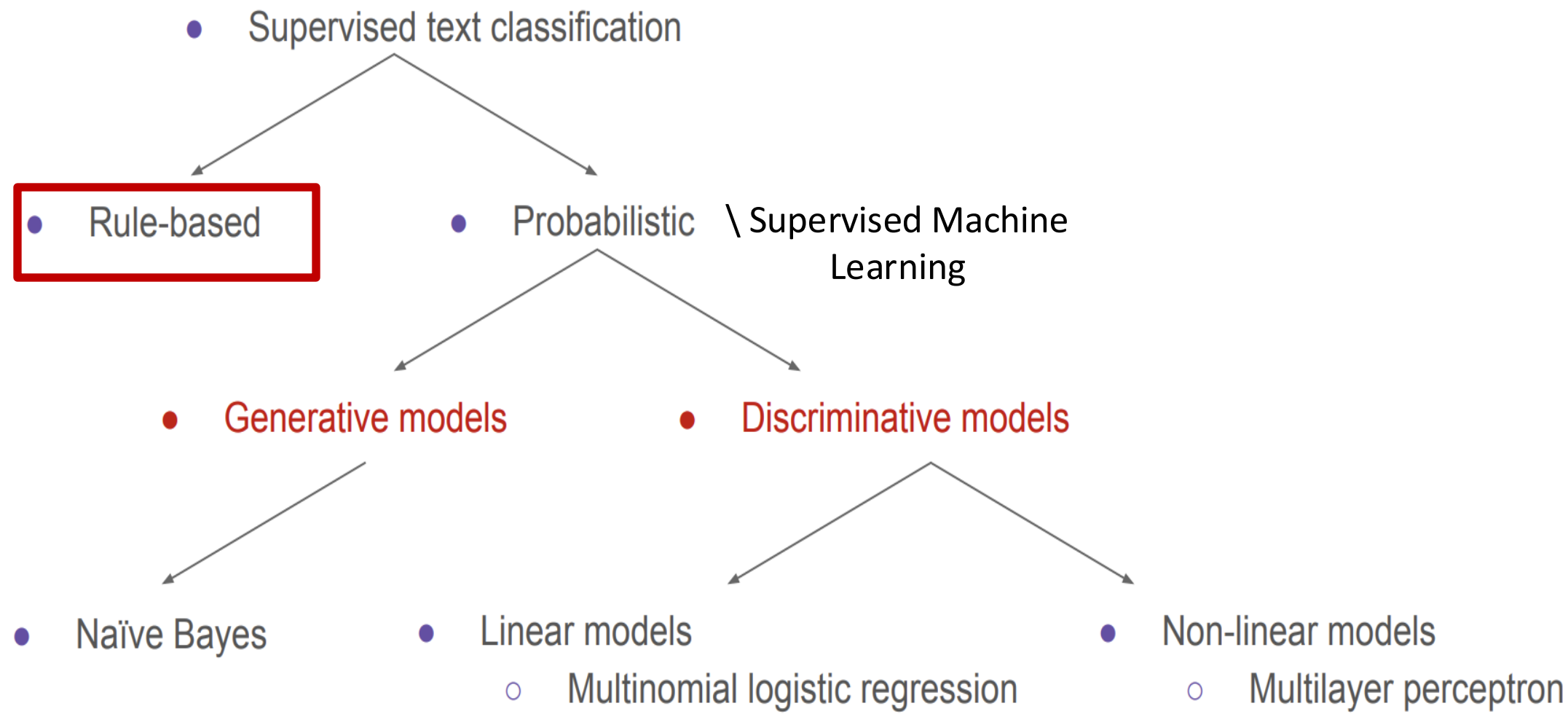
- **Output:**
  - a predicted class $\boldsymbol{c \in C}$

Supervised Text Classification Definition

1. ***Training:***
   - ***Input:***
     - 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 $\hat{y}: d \rightarrow c$
     - The hat or circumflex notation $\hat{y}$ is used to refer to an estimated or predicted value

2. ***Inference or Test:***
   - ***Input:*** a document d
   - ***Output:*** a class $\hat{y}$

Taxonomy of Text Classification Models



Rule-based Classifier

- Rule-based classifiers make the class decision based on **“if..else” rules**

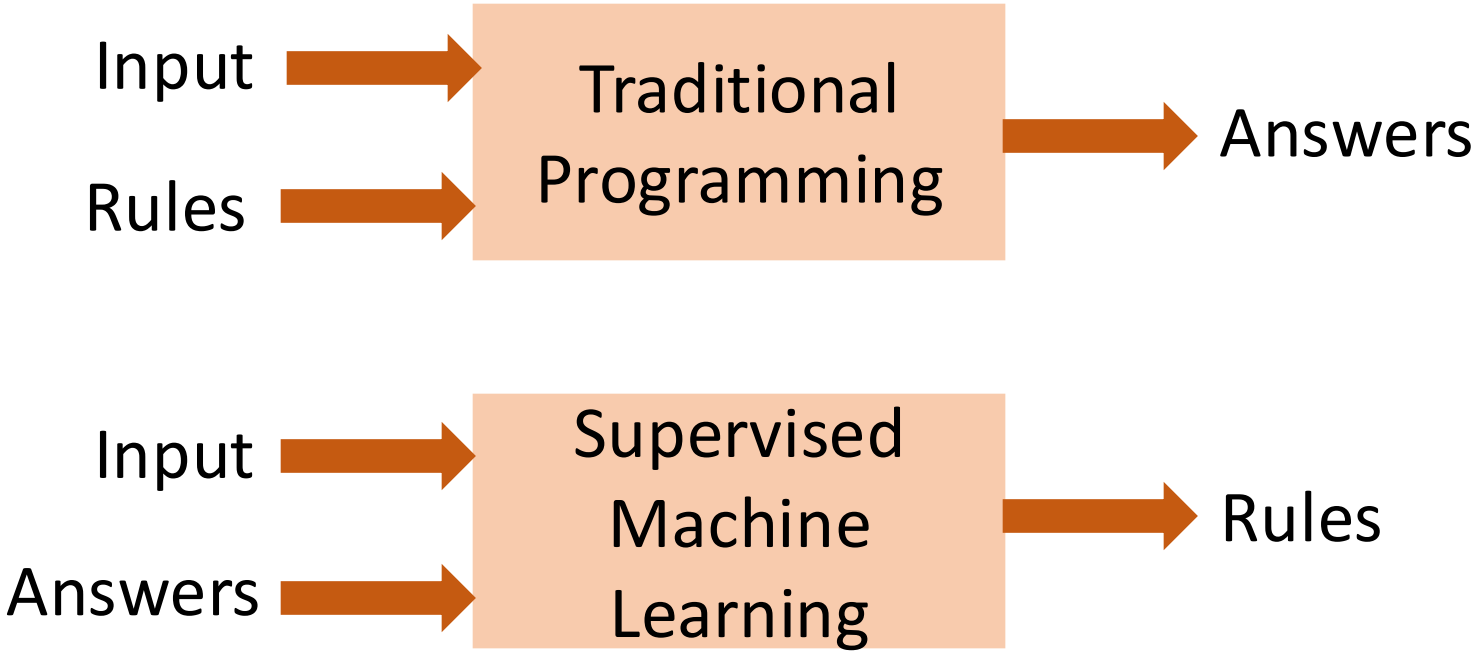
```
def classify_sentiment(document):
  for word in document:
    if word in {“good”, “wonderful”, “excellent”}:
      return 5
    if word in {“bad”, “awful”, “terrible”}:
      return 1
```

- These rules are easily interpretable and thus these classifiers are generally used to generate descriptive models.

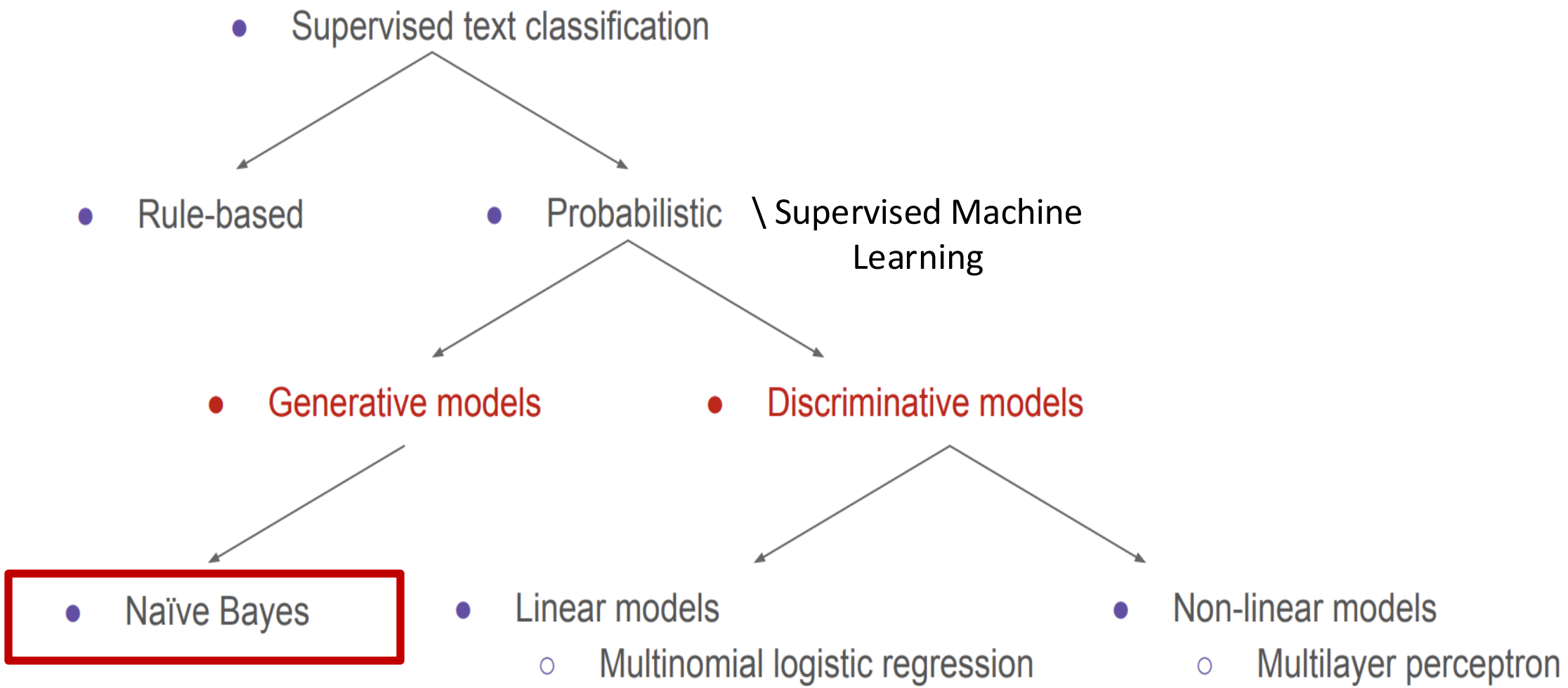
Rule-based Classifier: Challenges

- Rules may be hard to define (and some even unknown to us!)
  - **Sentiment**: Half submarine flick, half ghost story, all in one a criminally neglected film.
  - hard to identify a priori which words are informative (and what information they carry!)
  - **Sentiment:** It's not life-affirming, it's vulgar, it's mean, but I liked it.
  - word order matters, but hard to encode in rules!
- Expensive
- Not easily generalizable

Rule-based Vs Supervised Learning Classification



Taxonomy of Text Classification Models



Discriminative Vs Generative Classifiers

Quick Review on Probabilities

- Event space (e.g., $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$) – in this class, usually discrete
- Random variables (e.g., **X, Y**)
- Random variable X takes value **x**, **x**$\in \boldsymbol{\mathcal{X}}$ with probability with **p(X=x) or p(x)**

Quick Review on Probabilities

- **Conditional Probability** is the probability that one event will happen *given that* another action or event has already happened.

$$P(X, given\ Y) \text{ or } P(X|Y)$$

- **Joint probability** is the probability of event Y occurring at the same time that event X occurs.

$$P(X \cap Y) = P(X|Y) \times P(Y)$$

Quick Review on Probabilities

- **If X and Y are independent variables:**

$$P(X \cap Y) = P(X).P(Y)$$

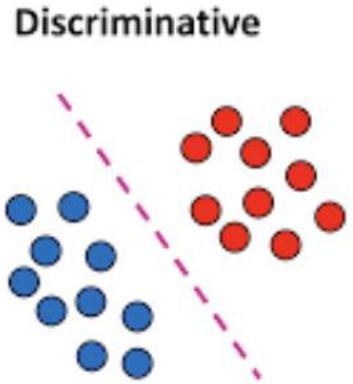
- **Otherwise:**

$$P(X \cap Y) = P(Y|X).p(X) = P(X|Y).p(Y)$$

Discriminative Vs ***Generative*** Classifiers

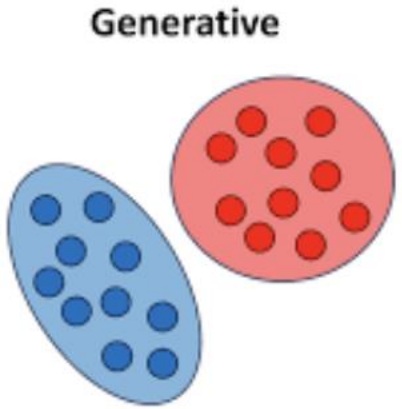
1. *Discriminative*

- ***Learns the decision boundary between classes***
- ***Focuses only on separating classes, not on how the data was generated.***
- Models the **conditional probability** $P(Y \mid X)$ – how likely a label Y is given features X
- Usually achieves **higher classification accuracy** when lots of labeled data are available



2. *Generative*

- ***Learns how the data is generated for each class.***
- Can **generate new samples** similar to the training data
- Models the **joint probability** $P(X \cap Y)=P(X|Y)P(Y)$.
- Generative models orks better with **small datasets** or **missing labels** since it captures more structure.



Source

Generative and ***Discriminative*** Classifiers

Suppose we're distinguishing cat from dog images

imagenet

imagenet

*How a **Generative** Classifier Works?*

Objective: Understand how data (pictures of dogs and cats) is created for each class:

- Learn the ***joint probability*** distribution $\boldsymbol{P(X, Y)}$.

- **Step: 1** Learn what dog pictures look like and what cat pictures look like by observing patterns in the data:
- **Learn the Likelihood** ($\boldsymbol{P(X \mid Y = dog)}$ and $\boldsymbol{P(X \mid Y = cat)}$.)

- **Step 2:** Learn how likely it is to see a dog or a cat in general.
- **Learn the prior probabilities** of each class $\boldsymbol{P(Y = dog)}$ and $\boldsymbol{P(Y = cat)}$

- **Step 3**: For a new picture, compare how similar it is to the learned patterns of both dog and cat images.
- **Compare** $\boldsymbol{P(X \mid Y = dog)P(Y = dog)}$ vs $\boldsymbol{P(X \mid Y = cat)P(Y = cat)}$

- **Step 4**: Classify the picture based on which class (dog or cat) is more likely to have generated the image.

Generative Classifier:

- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image:
    - how cat-y is this image?

Also build a model for dog images

Now given a new image:

Run both models and see which one fits better

*How a **Discriminative** Classifier Works?*

Objective: Directly learn how to separate dog pictures from cat pictures by learning ***the conditional probability*** $\boldsymbol{P(Y \mid X)}$, directly mapping input to a label.

- **Step 1**: Focus on learning the differences between dog and cat pictures/features (e.g., shape, fur, whiskers, etc.).

- **Step 2**: Learn a decision boundary that separates dog images from cat images based on the feature differences.

- **Step 3**: For a new image, predict $\boldsymbol{P(Y = dog \mid X)}$ and $\boldsymbol{P(Y = cat \mid X)}$, meaning check which side of the boundary the picture falls on.

- **Step 4**: Classify the image based on the highest probability prediction.

Discriminative Classifier

Just try to distinguish dogs from cats

Oh look, dogs have collars! Let's ignore everything else

Examples of ***Generative*** and ***Discriminative*** Classifiers

1. Generative Models

- **Naïve Bayes**
- Bayesian networks
- Hidden Markov Models (HMMs)
- Generative Adversarial Networks (GANs)
- Autoregressive Model
- ….

2. Discriminative Models

- Logistic regression
- Support Vector Machine (SVMs)
- Traditional neural networks
- K-Nearest neighbors
- Decision Trees and Random Forest
- ….

Naïve Bayes Classifier

Text Classification Steps

- **A Supervised Text Classification involves the following main Steps:**

1. Gathering and annotating data
2. Text Pre-processing (tokenization, stemming, lemmatization, etc…)
3. Text Representation (Features extraction)
4. Choosing and Training the Machine Learning Model
5. Testing the Machine Learning Model
6. Evaluating the model

Text Representation (Features Extraction)

- **Bag-of-Words (BOW)**
  - Easy, no effort required
  - Variable size, ignores sentential structure
- **Hand-crafted features**
  - Full control, class-specific features
  - Over-specific, incomplete
- **Learned feature representations**
  - Can learn to contain all relevant information
  - Needs to be learned

Generative Model: Naïve Bayes Classifier

- **Naive Bayes Intuition**
- Simple ("naive") classification method based on Bayes rule

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

- Relies on very simple representation of document
  - **Bag-of-words representations**

Bag-of-words Text Representation

- Given a document ***d*** (e.g., a movie review) – how to represent ***d***?

Chapter B

Figure 4.1 Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the *bag of words* assumption) and we make use of the frequency of each word.

Naïve Bayes Classifier

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

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

Naïve Bayes Classifier

"Likelihood" "Prior"

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

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

Document *d* represented by words $x_1, .. x_n$

Multinomial Naive Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

- **Bag of Words assumption**: Assume position doesn't matter
- **Conditional Independence (naive Bayes assumption)**: Assume the feature probabilities $\boldsymbol{P(x_i|c_j)}$ are independent given the class *c.*

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \cdot P(x_2 \mid c) \cdot P(x_3 \mid c) \cdot \ldots \cdot P(x_n \mid c)$$

Multinomial Naive Bayes Independence Assumptions

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

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

positions ← all word positions in test document

Problems with multiplying lots of probs

- There's a problem with this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

- Multiplying lots of probabilities can result in floating-point underflow!
- .0006 * .0007 * .0009 * .01 * .5 * .000008….
- Idea: Use logs, because log(*ab*) = log(*a*) + log(*b*)
- We'll sum logs of probabilities instead of multiplying probabilities!

Underflow prevention: log space

- Instead of this: $$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$
- We obtain this: $$c_{\mathrm{NB}} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \mathrm{positions}} \log P(x_i | c_j) \right]$$
- Notes:
  1) Taking log doesn't change the ranking of classes!
     The class with highest probability also has highest log probability!
  2) It's a linear model:
     Just a max of a sum of weights: a **linear** function of the inputs
     So naive bayes is a **linear classifier**

Multinomial Naïve Bayes: **Learning**

- How do we learn (train) the Naïve Bayes model?
- We learn $P(c_j)$ and $P(x_i|c_j)$ from training (labeled) data

$$c_{\text{NB}} = \underset{c_j \in C}{\text{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i|c_j) \right]$$

Multinomial Naïve Bayes: **Learning**

- For the class prior P($\boldsymbol{c_j}$) we ask what percentage of the documents in our training set are in each class $\boldsymbol{c_j}$

$$\widehat{\boldsymbol{P}}(\boldsymbol{c_j}) = \frac{\boldsymbol{N_{c_j}}}{\boldsymbol{N_{total}}}$$

$\boldsymbol{N_{c_j}}$: the number of documents in our training data with class $\boldsymbol{c}$

$\boldsymbol{N_{total}}$ be the total number of documents.

Multinomial Naïve Bayes: **Learning**

$$\widehat{P}(x_i|c_j) = \frac{count(x_i, c_j)}{\sum_{x \in V} count(x, c_j)}$$

fraction of times word x_i appears among all words in documents of class c_j

Here the vocabulary V consists of the union of all the word types in all classes, not just the words in one class c.

- Create mega-document for class *j* by concatenating all docs in this class
  - Use frequency of x_i in mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word "fantastic" and classified in the class positive?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

$$\widehat{\boldsymbol{P}}(\boldsymbol{x_i}|\boldsymbol{c_j}) = \frac{\boldsymbol{count}(\boldsymbol{x_i}, \boldsymbol{c_j}) + \mathbf{1}}{\sum_{\boldsymbol{x}\in\boldsymbol{V}}(\boldsymbol{count}(\boldsymbol{x}, \boldsymbol{c_j}) + \mathbf{1})}$$

$$= \frac{count(x_i, c_j) + 1}{\sum_{x\in V} count(x, c_j) + |V|}$$

Multinomial Naïve Bayes: **Learning**

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the +1 to +α for loglikelihood counts in training.

Chapter B

Unknown Words

- What about unknown words
  - that appear in our test data
  - but not in our training data or vocabulary?
- We ignore them
  - Remove them from the test document!
  - Pretend they weren't there!
  - Don't include any probability for them at all!
- Why don't we build an unknown word model?
  - It doesn't help: knowing which class has more unknown words is not generally helpful!

Stop Words

- Some systems ignore stop words
  - Stop words: very frequent words like the and a.
  - Sort the vocabulary by word frequency in training set
  - Call the top 10 or 50 words the stopword list.
  - Remove all stop words from both training and test sets
- But removing stop words doesn't usually help
- So in practice most NB algorithms use all words and don't use stopword lists

Summary: naïve Bayes is not so naïve

- Naïve Bayes is a probabilistic model
  - Naïve because it assumes features are independent of each other for a class
  - Very fast, low storage requirements

- Very good in domains with many equally important features
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification

But we will see other classifiers that give better accuracy

Reading

- Please refer to this document : Chapter B
- https://scikit-learn.org/stable/modules/naive_bayes.html
- https://www.lexalytics.com/blog/machine-learning-natural-language-processing/
- https://www.spiceworks.com/tech/artificial-intelligence/articles/top-ml-algorithms/

Exercise

Problem: You are given a small dataset of text messages labeled as **positive** or **negative** sentiment. Your task is to classify a new message into either "positive" or "negative" sentiment using the **Naive Bayes** approach.

Message	**Sentiment**
I love this product	Positive
This is the best day of my life	Positive
I hate this	Negative
This is terrible	Negative
Worst experience ever	Negative
Love the features	Positive

Exercise

1. Create a vocabulary of the unique words from the dataset. For each word, note how often it appears in both **positive** and **negative** sentiment messages.
2. Calculate the prior probabilities of the **positive** and **negative** sentiment classes.
3. Using **Laplace Smoothing**, calculate the likelihood of each word occurring in both positive and negative messages.
4. Classify the following message using the Naive Bayes approach: ***"I love this product"***
5. Explain how Laplace smoothing helps to handle unseen words in text classification.
6. What are the limitations of the Naive Bayes assumption of conditional independence between features in the context of text classification?

How do we evaluate our Classifier?

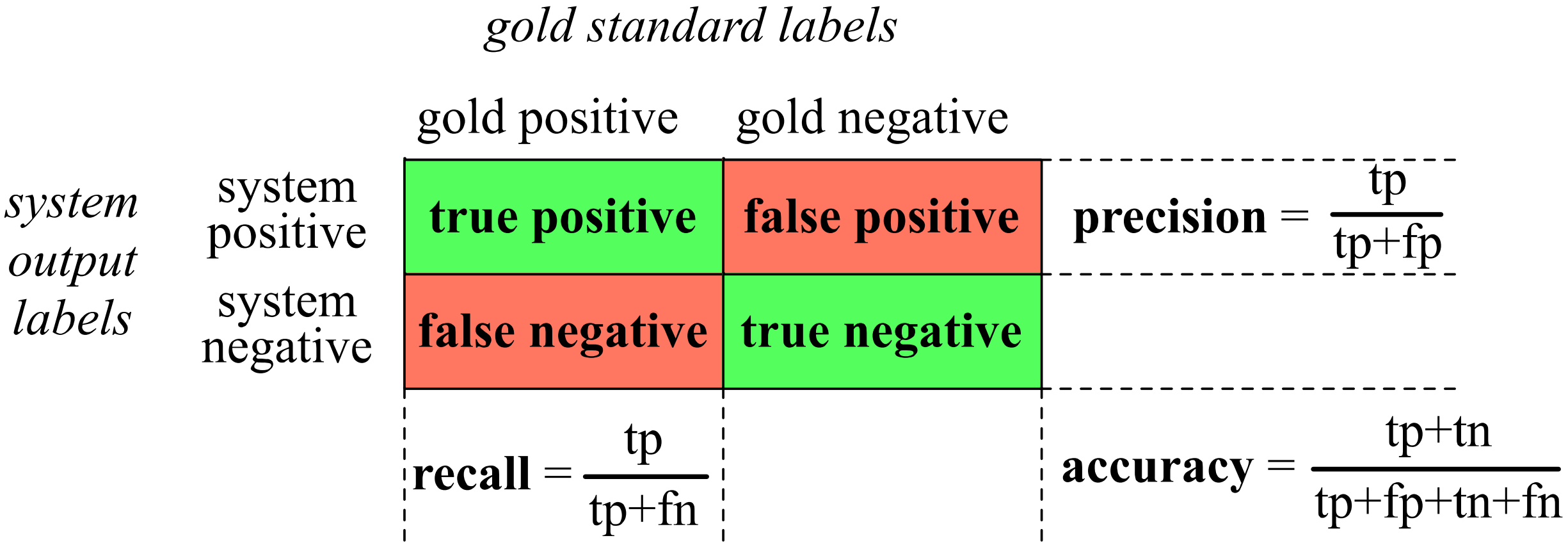
Example: Sentiment Analysis Classifier

- Let's consider just binary text classification tasks. Imagine you are the CEO of Shoes Company. You want to know what people are saying about your products.

- So you build a tweet detector
  - **Positive class**: tweets about wonderful shoes
  - **Negative class**: all other tweets

Confusion Matrix

- Confusion Matrix a.k.a Contingency table: model's predictions are compared to the correct results



Evaluation metric: Accuracy

- Why don't we use **accuracy** as our metric?
- Imagine we saw 1 million tweets
  - 100 of them are **Positive**.
  - 999,900 are **Negative**.

- We could build a dumb classifier that just labels every tweet ”***negative***”:
  - It would get 99.99% accuracy!!! Wow!!!!
  - But useless! Doesn't return the comments we are looking for!
  - That's why we use **precision** and **recall** instead

Evaluation metric: Precision

- **Precision (P)** measures the percentage of the items that the system detected (i.e., the system labeled as positive) that are in fact positive (i.e., are positive according to the human gold labels):

$$\textbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Evaluation metric: Recall

- **Recall (R)** measures the percentage of items actually present in the input that were correctly identified by the system

$$\textbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Evaluation metric: F-measure

- **F-measure:** a single number that combines Precision and Recall:

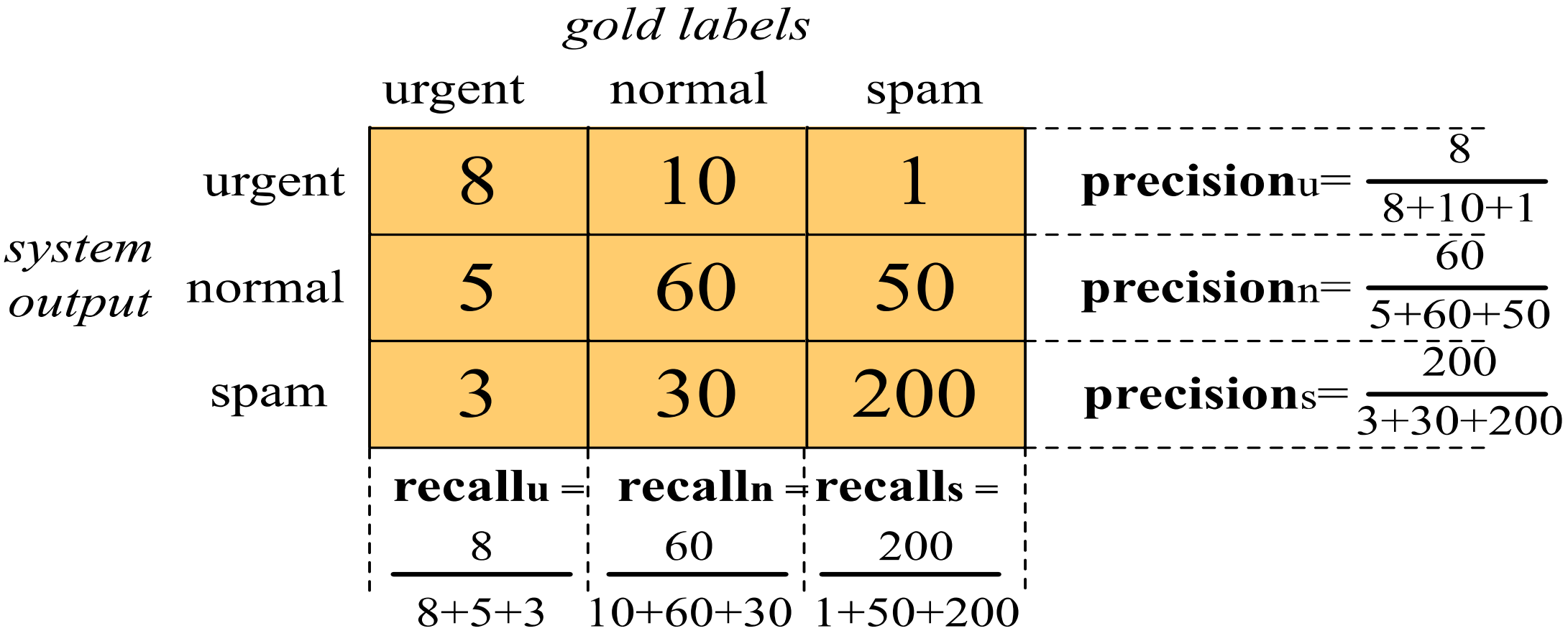
$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- The β parameter differentially weights the importance of recall and precision, based perhaps on the needs of an application
- We almost always use balanced F_1 (i.e., $\beta = 1$)

$$\mathrm{F}_1 = \frac{2PR}{P + R}$$

Evaluation with more than two classes

Evaluation with more than two classes:
Confusion Matrix for 3-class classification



Micro- vs. macro-averaging

- **Macro-averaging**:
  - Compute the performance for each class, and then average over classes
- **Micro-averaging**:
  - Collect decisions for all classes into one confusion matrix
  - Compute precision and recall from that table.

Micro- vs. macro-averaging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

Overfitting

- **Overfitting** refers to the case when a model is so specific to the data on which it was trained that it is no longer applicable to different datasets

- In situations where your training error is low but your test error is high, you've likely overfit your model.

- The model would not be able to generalize the data point in the test data set to predict the outcome accurately.

- A good model should be able to generalize

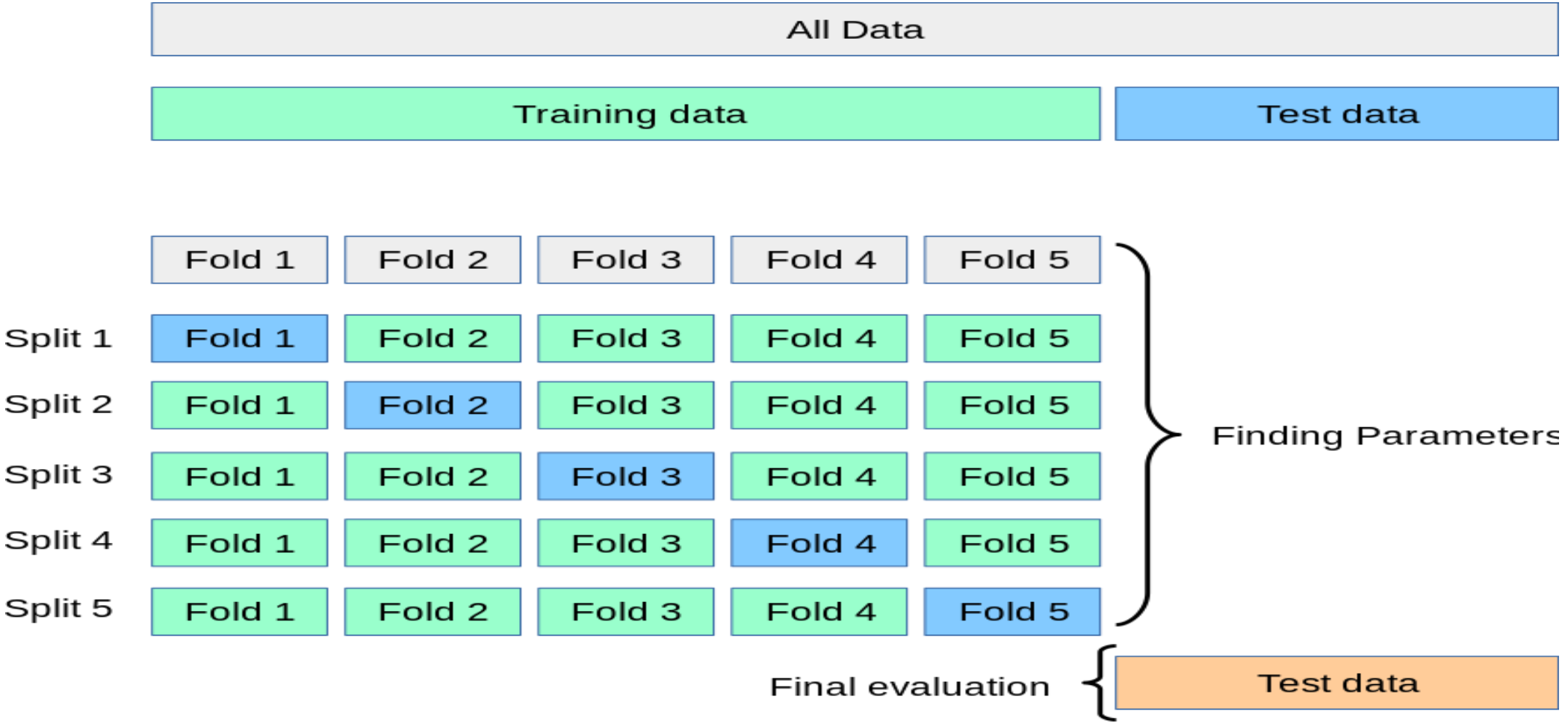
Training| Development| Test Sets

- Train on training set, tune on devset, report on testset
  - This avoids overfitting (‘tuning to the test set’)
  - But paradox: want as much data as possible for training, and as much for dev; how to split?

K-fold Cross Validation

- Divide the training data into $\boldsymbol{k}$ folds (e.g., $\boldsymbol{k = 5}$)
- Repeat $\boldsymbol{k}$ times: train on $\boldsymbol{k - 1}$ folds and test on the holdout fold, cyclically
- The performance measure reported by the $\boldsymbol{k}$ −fold cross validation is then the average of the values computed in the loop.

K-fold Cross Validation



Source

Regularization

- A solution for overfitting
- Add a regularization term $R(\theta)$ to the loss function (for now written as maximizing logprob rather than minimizing loss)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta)$$

- Idea: choose an $R(\theta)$ that penalizes large weights
  - fitting the data well with lots of big weights not as good as fitting the data a little less well, with small weights

L2 Regularization (= ridge regression)

- The sum of the squares of the weights
- The name is because this is the (square of the) **L2 norm** $||\theta||_2$, = **Euclidean distance** of θ to the origin.

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2$$

- L2 regularized objective function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} \theta_j^2$$

L1 Regularization (= lasso regression)

- The sum of the (absolute value of the) weights
- Named after the **L1 norm** $||W||_1$, = sum of the absolute values of the weights, = **Manhattan distance**

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i|$$

- L1 regularized objective function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j|$$

Statistical Significance Testing

- How do we know if one classifier is better than another?

https://www.investopedia.com/terms/s/statistically_significant.asp

Reading

- https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/
- https://scikit-learn.org/stable/modules/cross_validation.html

Next Class

- **Lecture 6: Text Classification (Part 2)**