

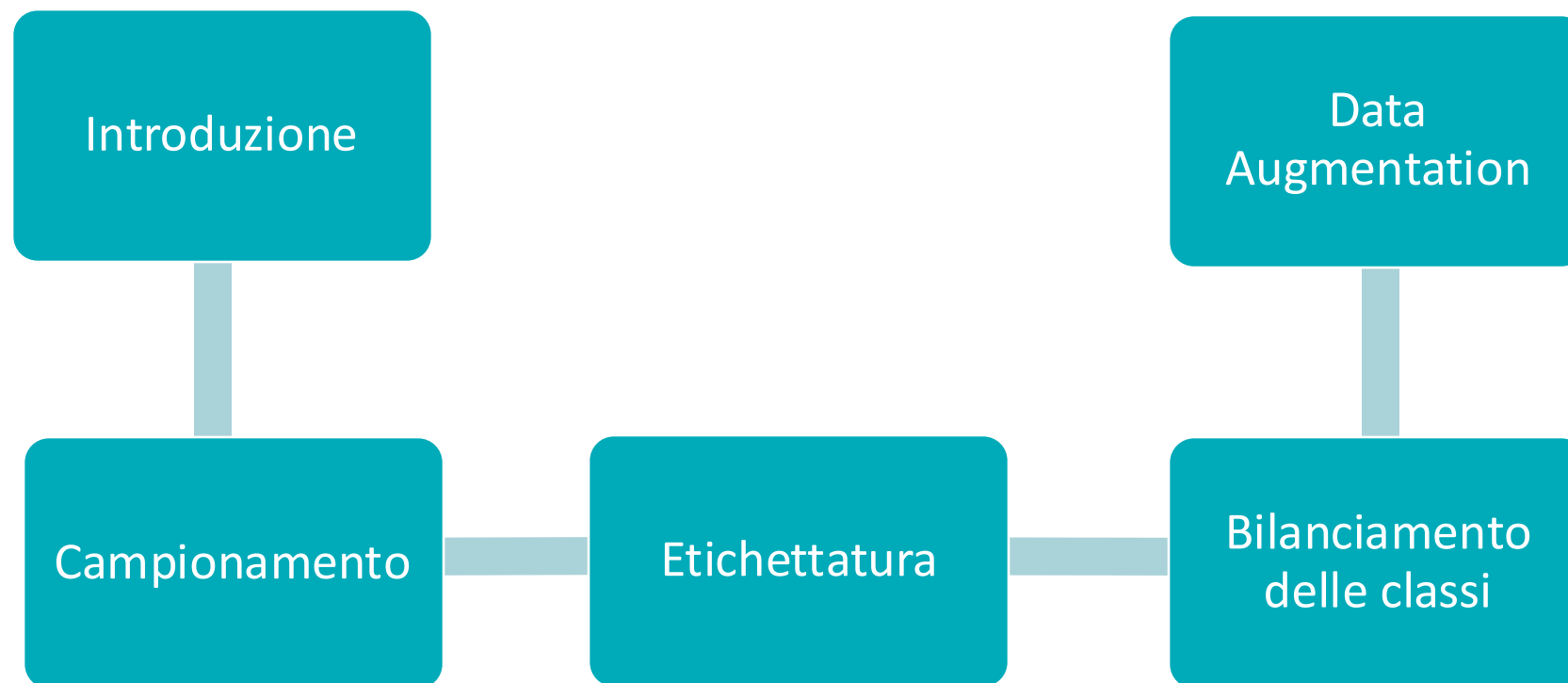
# Machine Learning

## 3 – Training Data



# Outline

---



# Introduzione

# Introduzione

---

- Uno degli aspetti fondamentali nella costruzione di un modello di Machine Learning è strettamente legato ai dati utilizzati per la fase di addestramento del modello.
  - Avere **dati puliti e di alta qualità** permette di addestrare modelli ad alte performance.
  - Diventa importante, quindi, costruire dataset adatti al dominio in cui si vuole lavorare.

# Alcune problematiche comuni

- Molto spesso, i dati possono essere:
  - **Complessi**
  - **Disordinati**
  - **Protetti**
  - **Potenzialmente pericolosi**
  - ...
- Inoltre, i dati sono pieni di potenziali bias
  - Per esempio, dati storici possono **implicitamente avere bias** legati ai comportamenti umani in determinati periodi storici (razzismo, sessismo, etc)
- Un **errato pre-processing** di questi dati influenza fortemente i modelli di ML e porta molto facilmente al fallimento di interi progetti.

# Esempio di modelli con bias

- nel 2018, un software di riconoscimento facciale di un'importante azienda di tecnologia ha dimostrato di avere difficoltà a riconoscere correttamente i volti delle persone di colore, classificandoli erroneamente come "oggetti" o identificandoli in modo errato.
- Ciò era dovuto al fatto che, in fase di addestramento, i dati a disposizione contenevano principalmente volti di persone bianche e pochi volti di persone di colore.
- In questo caso, quindi, il modello aveva acquisito involontariamente un bias razziale.

# Campionamento

# Il campionamento dei dati

- Il campionamento dei dati è un passo molto importante nella definizione di un modello di ML
- Può essere fatto nei diversi step del ciclo di vita di un progetto di ML:
  - **Creazione** dei dati di addestramento,
  - Divisione di questi dati in training, validation e test set,
  - **Definizione ed individuazione** di tutti i possibili eventi che possono accadere durante la messa in opera di un sistema di ML.



# Metodologie di campionamento

- Esistono moltissime tecniche di campionamento. In genere, vengono divise in due grandi famiglie:
  - **Campionamento non probabilistico**
  - **Campionamento casuale**
- Conoscere caratteristiche ed applicazioni di queste metodologie è importante per:
  - **Evitare** la presenza di potenziali bias nei dati,
  - **Migliorare** la qualità e l'efficienza dei dati.

# Campionamento non probabilistico

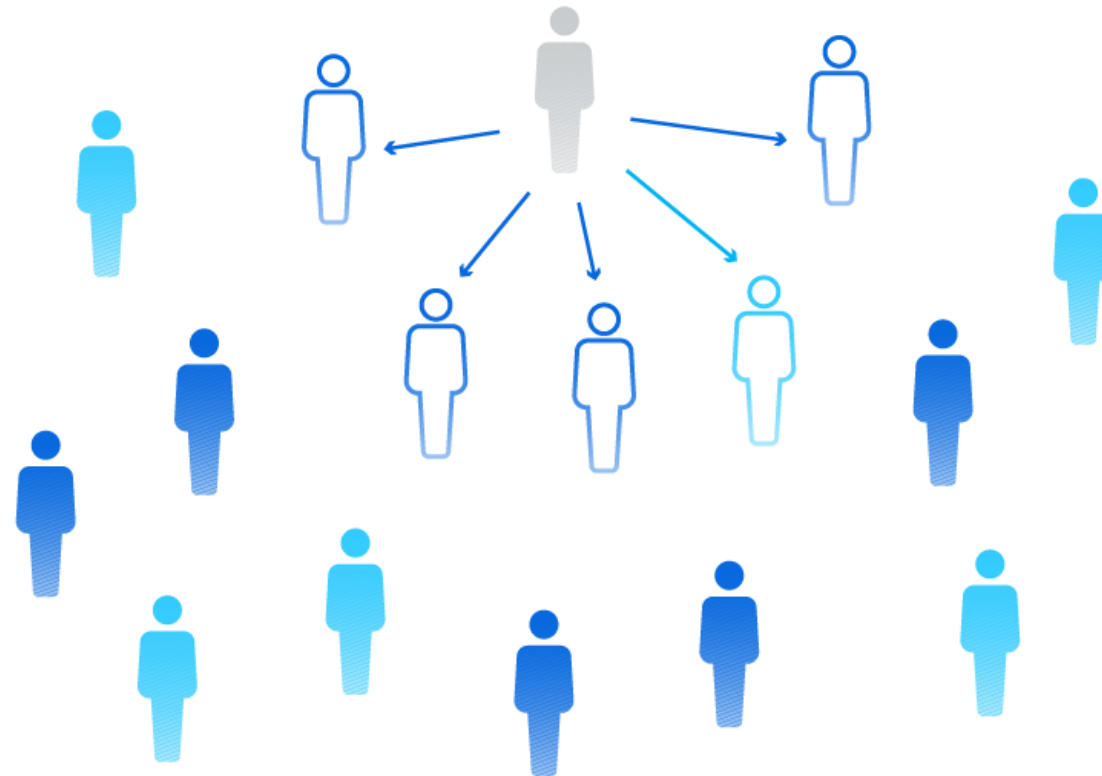
- Per campionamento non probabilistico si intendono tutte quelle tecniche di selezione dei dati che non si basano sull'utilizzo di criteri probabilistici.
- Esistono molte metodologie di campionamento di questo tipo:
  - **Campionamento di convenienza** (Convenience sampling)
  - **Campionamento a valanga o a palla di neve** (Snowball sampling)
  - **Campionamento per giudizio** (Judgment sampling)
  - **Campionamento per quote** (Quota sampling)
- È importante sapere che, però, i campioni selezionati con queste tecniche **non sono rappresentativi** dei dati del mondo reale (bias di campionamento).

# Campionamento di convenienza

- Questa tecnica di campionamento prevede che i campioni dei dati vengano selezionati in base alla loro **disponibilità**
  - Invece di scegliere i campioni in maniera casuale, vengono scelti i campioni **più facili da acquisire**
  - Metodologia **semplice, veloce e poco costosa**
- Viene comunemente utilizzato in situazioni in cui si ha una grande popolazione e testare tutti i membri non sarebbe possibile
  - Problemi a livello di tempo
  - Problemi di risorse
- **Problemi:** Correttezza dei dati e potenziale presenza di bias

# Campionamento di convenienza

## Convenience sample



# Esempio di campionamento di convenienza

- Un ricercatore vuole condurre uno studio sulle esperienze dei sopravvissuti al cancro.
- Sfruttando il campionamento di convenienza, la raccolta dei dati avverrà attraverso l'uso di:
  - Gruppi sui social network
  - Conoscenze personali di persone che hanno avuto tale problematica
  - Reti sociali di supporto per il cancro
- L'utilizzo di questi strumenti è molto semplice e permette di ottenere le informazioni necessarie allo studio partendo da ciò che è più vicino al ricercatore

# Pro e contro del campionamento di convenienza

## PRO

- Rapida raccolta dei dati
- Creazione di campioni poco costosi
- Facilità di ricerca
- Basso costo
- Meno regole da seguire

## CONTRO

- Bias di campionamento
- Mancanza di varietà
- Validità limitata
- Difficoltà nell'individuare errori

# Campionamento a valanga (palla di neve)

- Tecnica di campionamento che prevede la scelta dei campioni in base ai campioni **già esistenti**
  - Prima si definisce una piccola popolazione di campioni già noti
  - Da questi, si vanno ad identificare altri campioni che dovrebbero essere inclusi nella popolazione
- Questa tecnica viene utilizzata maggiormente in situazioni in cui:
  - Non si ha accesso a tutta la popolazione
  - È difficile identificare i campioni da scegliere
- **Problemi:** bias da rete sociale e limitata rappresentatività

# Campionamento a valanga





# Esempio di campionamento a valanga

- Supponiamo di voler studiare le abitudini di acquisto dei clienti per un grande centro commerciale.
- Tuttavia, è difficile ottenere un elenco completo di tutti i clienti.

Utilizziamo questa tecnica per definire il dataset:

- Iniziamo **selezionando casualmente  $k$  persone** presenti nel centro commerciale e chiediamo di partecipare al nostro sondaggio.
- Dopo aver ricavato dati da questi, chiediamo loro di **segnalarci altri clienti che conoscono** e che potrebbero fornirci altri dati.
- Ripetiamo questi passi e, **proprio come una palla di neve**, il dataset diventerà sempre più grande.

# Pro e contro del campionamento a valanga

## PRO

- Accesso a popolazioni nascoste o difficili da raggiungere
- Riduzione costi e tempi
- Identificazione di nuovi elementi che possono essere stati trascurati con altri metodi

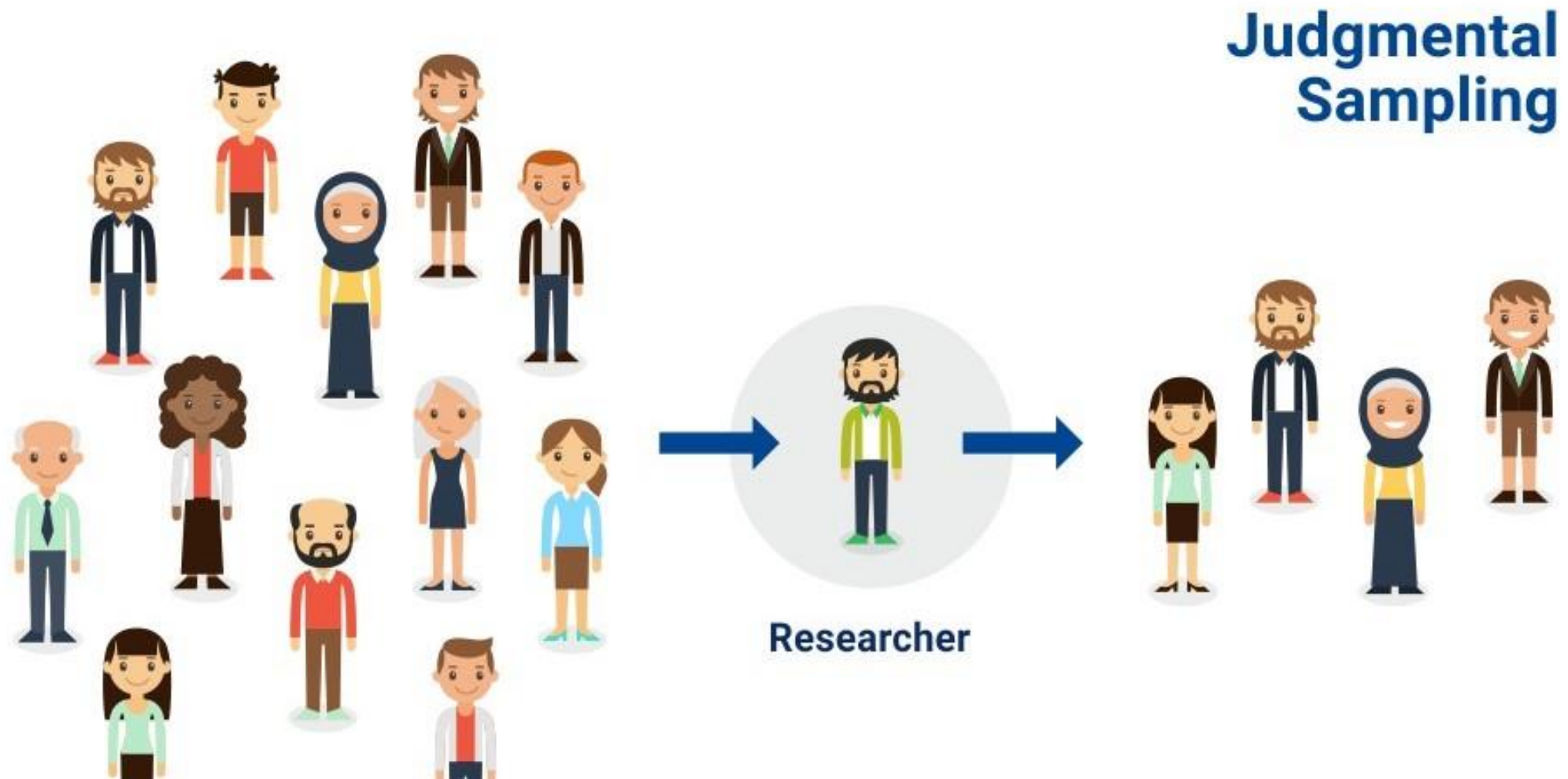
## CONTRO

- Bias di selezione
- Campioni non rappresentativi
- Difficoltà nel valutare gli errori di campionamento
- Dipendenza dai campioni scelti nelle fasi iniziali

# Campionamento per giudizio

- Tecnica di campionamento molto differente da quelle viste in precedenza.
  - In questo caso, i campioni non vengono scelti in maniera casuale
  - Bensì, vengono selezionati dai ricercatori stessi o da esperti del dominio
- Questa tecnica viene utilizzata quando:
  - I ricercatori ritengono che determinati elementi siano particolarmente rilevanti o significativi per lo studio da fare.
  - Studi qualitativi in cui la qualità dei dati è più importante della rappresentatività statistica
- **Problemi:** Poca precisione statistica e difficoltà nel replicare gli esperimenti in contesti diversi.

# Campionamento per giudizio



# Esempio di campionamento per giudizio

- Supponiamo di voler studiare il livello di soddisfazione dei clienti di un ristorante.
- Abbiamo a disposizione un set di 100 clienti. Il ricercatore sceglie 20 clienti in base a criteri soggettivi:
  - Clienti che hanno scritto recensioni positive o negative
  - Clienti che hanno visitato il ristorante molto frequentemente in un determinato periodo di tempo
- Dopo aver scelto questi campioni, il ricercatore ottiene informazioni da essi e trae conclusioni sul livello di soddisfazione complessivo dei clienti.

# Pro e contro del campionamento per giudizio

## PRO

- Flessibilità nella selezione
- Utilizzo in contesti qualitativi
- Accesso ad informazioni dettagliate

## CONTRO

- Bias di selezione
- Campioni non rappresentativi
- Difficoltà nella replicabilità
- Difficoltà nel calcolo dell'errore di campionamento

# Campionamento per quote

- Questa tecnica di campionamento si basa sull'individuazione di **quote** per determinate porzioni dei dati
  - Quote definite sulla base **di caratteristiche specifiche dei dati**
  - Si stabiliscono queste quote e si selezionano i campioni in modo tale da soddisfarle
- Questa tecnica diviene molto utile in situazioni in cui:
  - Le risorse sono limitate e non possiamo usare altre tecniche
  - Si vogliono raccogliere dati per popolazioni rare o particolari
  - Si vuole svolgere una ricerca in domini specifici e per particolari task
- **Problemi:** potenziale presenza di bias e limitata rappresentatività statistica

# Campionamento per quote





# Esempio di campionamento per quote

- Supponiamo di voler condurre un sondaggio sull'opinione dei consumatori riguardo ad un nuovo prodotto.
- La popolazione è composta da persone di diverse età:
  - Persone giovani (18 – 30 anni)
  - Adulti (31 – 50 anni)
  - Anziani (oltre i 50 anni)
- Si stabiliscono delle quote e si selezionano i campioni in maniera non casuale
  - Per esempio, vogliamo ottenere i dati da 10 giovani, 20 adulti e 30 anziani

# Pro e contro del campionamento per quote

## PRO

- Rapidità nel raccogliere i dati
- Contenimento dei costi
- Non richiede particolari criteri di campionamento

## CONTRO

- Bias di selezione
- Difficoltà nell'ottenere una rappresentatività statistica
- Limitazioni nella generalizzabilità dei risultati

# Oltre il campionamento non probabilistico

- Finora abbiamo visto diverse tecniche di campionamento non probabilistico
  - Queste tecniche sono molto utili per costruire un dataset iniziale per l'addestramento di un modello di ML
- Tuttavia, per ottenere modelli più affidabili, queste tecniche non sono le più indicate
  - La potenziale presenza di bias è un problema importante da trattare
  - È necessario che il dataset sia rappresentativo della realtà di interesse
- Possiamo risolvere queste problematiche attraverso l'utilizzo delle tecniche di **Campionamento casuale**.

# Campionamento casuale

- Con **Campionamento casuale** si intendono tutte quelle tecniche di campionamento dove la selezione dei dati avviene attraverso la definizione di criteri probabilistici.
  - L'esatto opposto del campionamento non probabilistico, dove non viene utilizzato alcun criterio di questo tipo
- Tra le tecniche più importanti possiamo trovare:
  - **Campionamento casuale semplice** (Simple Random Sampling)
  - **Campionamento stratificato** (Stratified Sampling)
  - **Campionamento pesato** (Weighted Sampling)
  - **Campionamento a serbatoio** (Reservoir Sampling)
  - **Campionamento per importanza** (Importance Sampling)

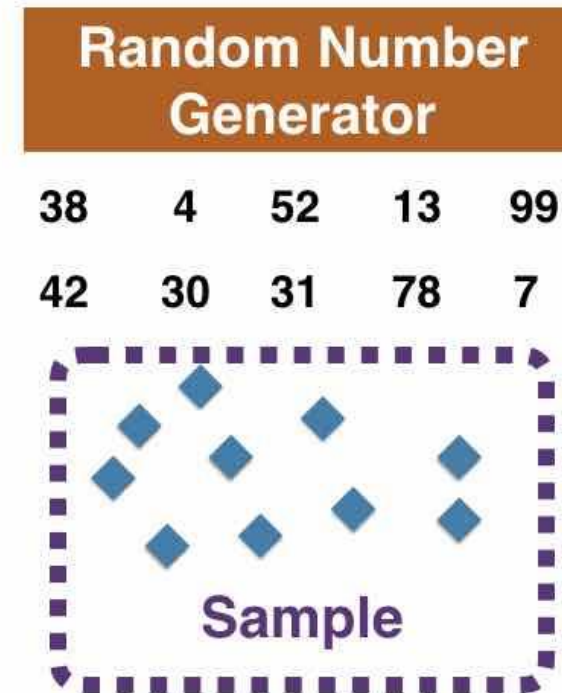
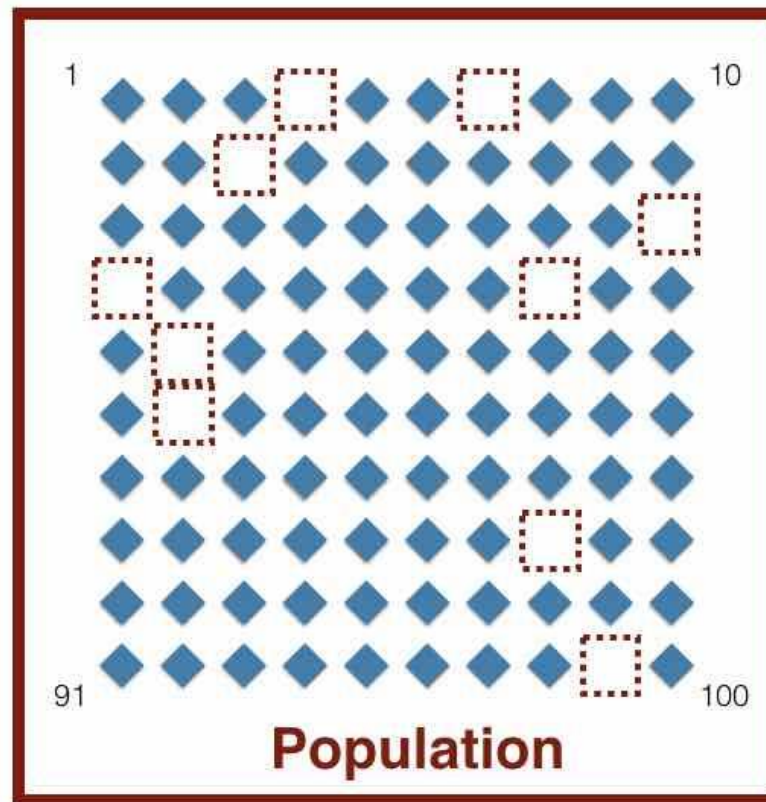
# Campionamento casuale semplice

- Questa tecnica può essere considerata la più semplice di questo tipo
  - La selezione dei campioni avviene assegnando uguale probabilità ad ogni elemento della popolazione,
  - Quindi, tutti i campioni hanno la stessa probabilità di essere scelti.
- Poco elaborata e molto semplice da implementare. Tuttavia, possono esserci problemi a livello di classi dei campioni
  - Per esempio, se assegno una probabilità di scelta pari a 0.01% e seleziono solo 1% della popolazione, potrebbe accadere che elementi di una classe rara non vengano scelti
  - In questo caso, il modello apprende solo le classi note, considerando le altre come inesistenti

# Campionamento casuale semplice

## Simple Random Sampling

*Every element has equal probability to be selected*



# Esempio di campionamento casuale semplice

- Supponiamo di voler acquisire informazioni sulle preferenze musicali degli studenti universitari.
- Abbiamo una popolazione di 1000 studenti e vogliamo sceglierne 100.
- Con questo tipo di campionamento, seguiamo i seguenti passi:
  - Assegniamo un identificativo ad ognuno dei 1000 studenti
  - Definiamo una probabilità di selezione uguale per ognuno di essi
  - Utilizziamo un estrattore di numeri casuali per definire i 100 campioni da scegliere

# Pro e contro del campionamento casuale semplice

## PRO

- Facile da implementare
- Semplicità nella definizione dei criteri di selezione

## CONTRO

- Classi rare possono non essere scelte
- Problematico con carenza di dati
- Non tiene conto delle correlazioni tra i dati
- Può essere costoso con popolazioni molto grandi



# Campionamento stratificato

- Questa tecnica cerca di risolvere i problemi del campionamento semplice
  - In questo caso, la popolazione viene inizialmente divisa in gruppi
  - Successivamente, si campiona da ogni gruppo in maniera separata ed indipendente
- Utilissimo quando si lavora con popolazioni:
  - eterogenee
  - che presentano diverse classi rare
  - con membri che presentano caratteristiche ben definite
- Permette di ottenere dataset bilanciati e ben rappresentativi della realtà

# Campionamento stratificato

## Stratified sampling



# Esempio di campionamento stratificato

- Supponiamo di voler condurre un'indagine sulla soddisfazione dei clienti di un grande centro commerciale. La popolazione è composta da clienti di diverse fasce di età: giovani (18-25 anni), adulti (26-40 anni) e anziani (oltre i 40 anni).
- Dividiamo la popolazione in gruppi, prendendo in considerazione l'età, e selezioniamo 50 campioni da ogni strato:
  - **Strato 1:** clienti giovani
  - **Strato 2:** adulti
  - **Strato 3:** anziani
- Per ogni strato, selezioniamo i 50 campioni applicando una selezione casuale

# Pro e contro del campionamento stratificato

## PRO

- Rappresentatività
- Precisione nel campionamento
- Efficienza
- Permette di fare confronti tra gruppi

## CONTRO

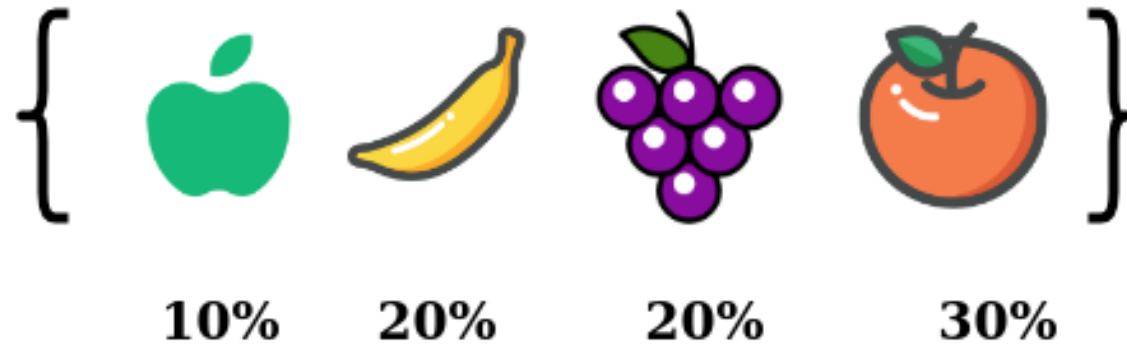
- Difficile scegliere il criterio di raggruppamento
- In alcuni casi, è impossibile dividere la popolazione in gruppi
- Complesso da implementare
- Costi maggiori
- Analisi molto più complesse

# Campionamento pesato

- Il campionamento pesato si basa sull'assegnazione di un **peso** ad ogni elemento/classe della popolazione.
  - Si definiscono dei pesi sulla base **della distribuzione della popolazione**
  - Il **peso rappresenta la probabilità** che quel campione venga scelto
  - Un **peso maggiore** equivale ad una **maggiore probabilità** di essere scelto
- Questa tecnica è molto efficace quando si lavora con popolazioni che:
  - presentano classi di dimensioni molto diverse tra loro
  - hanno sottopopolazioni rare
  - si vuole ottimizzare l'uso delle risorse a disposizione
- Questa tecnica, inoltre, permette di risolvere il problema del **bias di non risposta** (mancata partecipazione di alcune entità campionate)

# Campionamento pesato

## Weighted Sampling



*# Choose two items from the list such that 1, 2, 3, 4 each has  
# 20% chance of being selected, while 100 and 1000 each have only 10% chance.*

```
import random
```

```
random.choices(population=[1, 2, 3, 4, 100, 1000],  
               weights=[0.2, 0.2, 0.2, 0.2, 0.1, 0.1],  
               k=2)
```

*# This is equivalent to the following*

```
random.choices(population=[1, 1, 2, 2, 3, 3, 4, 4, 100, 1000],  
               k=2)
```

# Esempio di campionamento pesato

- Supponiamo di voler creare un dataset attingendo da una popolazione reale che presenta campioni classificati come rossi ed altri come blu.
- Sappiamo che, nella popolazione abbiamo una proporzione tra rossi e blu pari a 25/75.
- Inoltre, sappiamo che nel mondo reale, sia rossi che blu hanno la stessa probabilità di essere scelti.
- In questo caso, il campionamento pesato ci permette di ottenere un dataset equilibrato:
  - Possiamo assegnare peso 1 ad un esempio blu
  - Per i rossi, essendo un terzo di quelli blu, diamo un peso pari a 3, in modo tale da avere una probabilità di scelta maggiore

# Pro e contro del campionamento pesato

## PRO

- Rappresentatività
- Precisione nel campionamento
- Flessibilità del campione
- Permette di gestire situazioni in cui abbiamo carenza di dati

## CONTRO

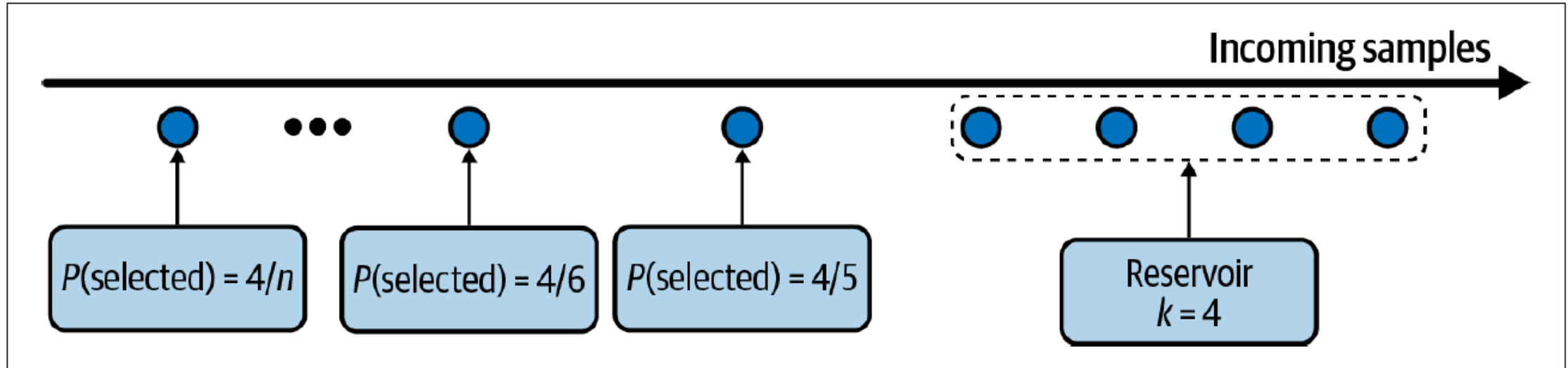
- Complesso calcolare i vari pesi
- Maggiore difficoltà nell'analisi dei risultati
- Possibile presenza di bias
- Gestione dei pesi molto complessa



# Campionamento a serbatoio

- Le tecniche viste finora non lavorano bene in situazioni in cui la popolazione cambia (ad esempio, in uno streaming di dati). Il campionamento a serbatoio ci permette di lavorare in questo tipo di situazioni.
  - Prevede la definizione di un **serbatoio** (un array, un set, etc)
  - Questa struttura viene inizialmente **riempita con un certo numero di elementi**
  - Man mano che altri elementi sono disponibili, gli elementi nel serbatoio vengono **rimpiazzati** secondo un **criterio predefinito**
- Con questa tecnica, è possibile fare un campionamento dove ogni elemento ha la stessa probabilità di essere selezionato, nonostante non si conoscano le dimensioni della popolazione.

# Campionamento a serbatoio



# Esempio di campionamento a serbatoio

- Supponiamo di voler definire un dataset di tweet provenienti da uno stream.
- Data la natura della sorgente, non possiamo conoscere quanti tweet abbiamo a disposizione e, ovviamente, non possiamo memorizzarli tutti in memoria.
  - Quindi, non conosciamo la probabilità di scelta di ogni tweet.
- Il campionamento a serbatoio ci viene in aiuto:
  - Inizialmente, inseriamo i primi  $k$  tweet nel nostro serbatoio (array o altro)
  - Per ogni  $n$ -imo elemento, generiamo un numero casuale  $i$ , tale che
$$1 < i < n$$
  - Se  $i$  è compreso tra  $1$  e  $k$ , allora **rimpiazziamo** l'elemento  $i$ -simo nel serbatoio con l'elemento  $n$ -simo. In caso contrario, non facciamo nulla.

# Pro e contro del campionamento a serbatoio

## PRO

- Adatto a popolazioni dinamiche
- Efficienza del campionamento
- Rappresentatività per streaming data
- Adatta a popolazioni di grandissime dimensioni

## CONTRO

- Complessità elevata
- Applicazione limitata
- Possibilità di avere distorsioni nella selezione dei campioni
- Difficile gestione dei criteri di selezione.

# Campionamento per importanza

- Una delle tecniche più conosciute ed importanti, non solo nel ML.
- Questa tecnica di campionamento viene utilizzata quando si vuole campionare una distribuzione sfruttandone un'altra più semplice da campionare
  - Immaginiamo di avere un campione  $x$  da una distribuzione  $P(x)$
  - Questa distribuzione è costosa, lenta o difficile da campionare
  - Abbiamo un'altra distribuzione  $Q(x)$  che è più facile da campionare
  - possiamo allora campionare  $x$  da  $Q(x)$  e pesarlo con  $\frac{P(x)}{Q(x)}$
- In questo caso,  $Q(x)$  è chiamata **proposal distribution** o **importance distribution**

# Esempio di campionamento per importanza

- L'esempio più classico è legato ai modelli di Reinforcement Learning basati su policy.
- Consideriamo il caso in cui si vuole aggiornare la policy del modello.
- Possiamo cercare di stimare i valori della nuova policy, ma calcolare il totale delle ricompense ottenute facendo un'azione può essere lento e costoso, in quanto si devono tener conto di tutti i possibili scenari.
- Tuttavia, se la nuova policy è molto simile alla vecchia, possiamo calcolare le ricompense totali della vecchia policy e ripesare queste ricompense secondo la nuova policy definita.

# Pro e contro del campionamento per importanza

## PRO

- Efficiente con distribuzioni difficili da campionare
- Adattabilità a diversi scenari di analisi
- Possibilità di estendere l'intervallo di campionamento per includere valori rari o di interesse

## CONTRO

- Difficoltà nella scelta della importance distribution
- Varianza delle stime
- Computazionalmente complesso
- Distorsione nelle stime se le distribuzioni sono troppo diverse tra loro.

# L'importanza di campionare i dati

- In molti casi reali, campionare i dati è necessario:
  - Spesso, non è possibile accedere a tutti i possibili dati disponibili nel mondo reale.
  - Oppure, si hanno a disposizione un'enorme quantità di dati e non è possibile processarli tutti (mancanza di risorse/tempo, etc).
- In questi casi, è fondamentale costruire dei sottoinsiemi di questi dati per procedere con l'addestramento/miglioramento di un modello di ML.
- Tuttavia, campionare i dati può essere utile anche per ridurre i costi e per velocizzare il completamento di un task



# Etichettatura

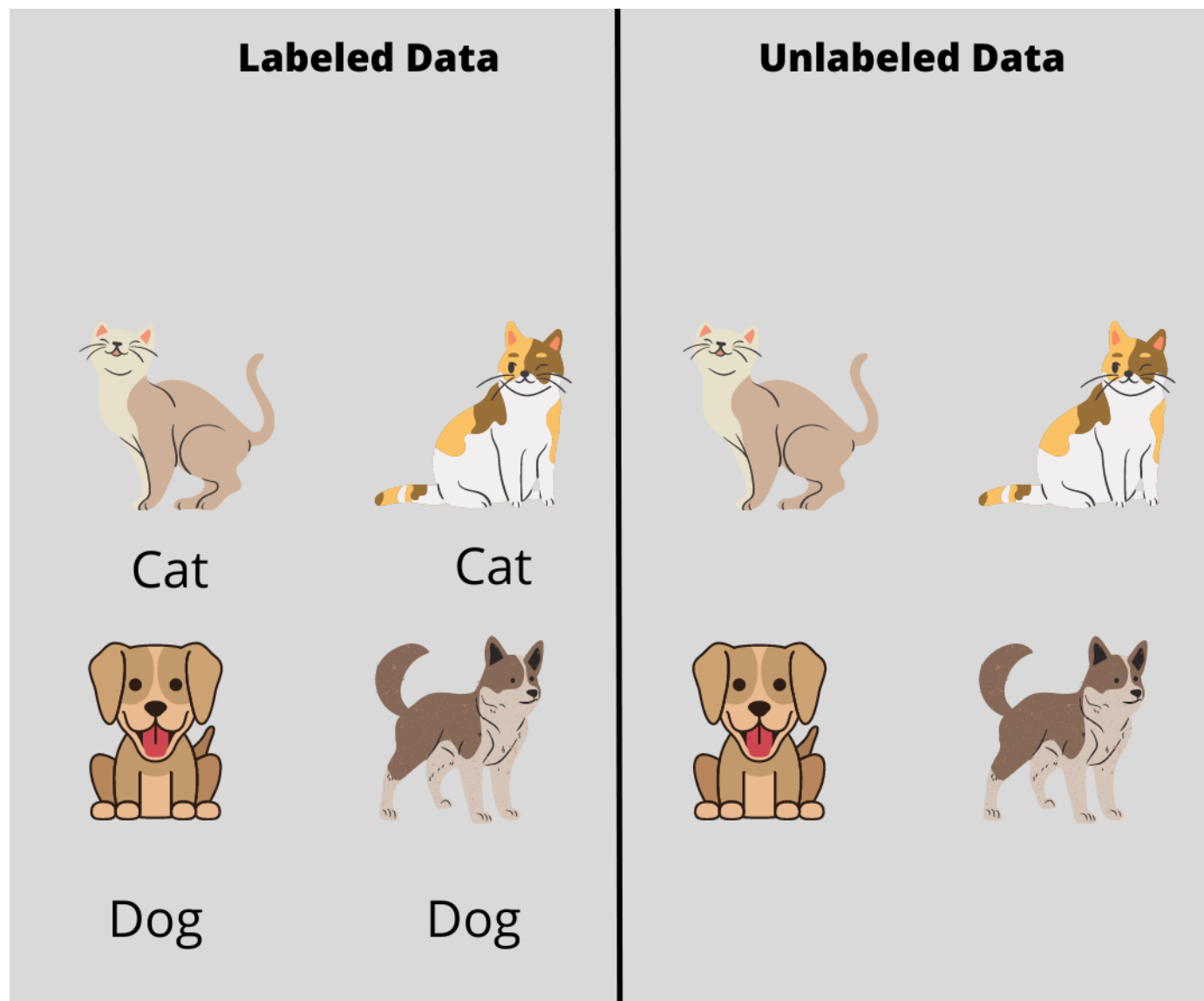
# Oltre al campionamento dei dati: l'etichettatura

- Con le tecniche di campionamento viste finora, siamo in grado di definire il nostro dataset di addestramento per il nostro modello di ML
- Ulteriori processi di data preparation dipendono dal tipo di addestramento:
  - Un modello non supervisionato ci permette di utilizzare questi dati direttamente
  - Un approccio supervisionato, invece, necessita di dati **etichettati**
- In quest'ultimo caso, la qualità e la quantità di dati di addestramento etichettati incidono moltissimo sulle performance del modello.
  - Il **processo di etichettatura**, quindi, gioca un ruolo fondamentale

# L'etichettatura dei dati

- Quando si parla di etichettatura dei dati, si fa riferimento al processo di **assegnazione di una o più etichette** ad ogni singolo elemento di un dataset
  - Per esempio, abbiamo un set di immagini ed assegniamo ad ognuna un'etichetta "*cane*" o "*gatto*" in base all'animale raffigurato in essa.
- Processo fondamentale per l'addestramento di modelli supervisionati
- Esistono dati già etichettati (**etichettatura naturale**) e dati che devono essere etichettati (**manualmente o con tecniche automatiche**)

# L'etichettatura dei dati



# L'etichettatura a mano

- L'etichettatura a mano è una tecnica con la quale si vanno a **definire manualmente le etichette** per ogni campione presente nel dataset.
  - Per esempio, vogliamo classificare 20 mail come “spam” o “non spam”.
  - Con questa tecnica, si esamina ogni mail singolarmente e si assegna un'etichetta sulla base di **una valutazione soggettiva**.
- L'utilizzo di questa tecnica è strettamente legata a **diverse problematiche**, come costi elevati, privacy, tempi troppo lunghi etc.

# Problemi dell'etichettatura a mano: i costi

- Uno dei problemi principali è sicuramente legato ai costi di un processo di etichettatura a mano
  - Considerando l'esempio delle mail di spam, etichettare 20 mail ha un costo relativamente basso
  - Etichettarne 20.000, invece, può essere molto costoso a causa del fatto che bisogna pagare un certo numero di annotatori per avere un risultato in tempi relativamente brevi
- Ovviamente, i costi aumentano all'aumentare dell'expertise degli annotatori ingaggiati per tale lavoro

# Problemi dell'etichettatura a mano: la privacy

- Un altro problema da tenere in considerazione è la **privacy dei dati**
  - Bisogna tener conto che **non tutti i dati** possono essere accessibili ad ogni persona
  - Esistono tipi di dati, come le informazioni personali di una persona, che sono **protetti da stretti vincoli di privacy**
- Considerando, per esempio, i dati medici di una serie di pazienti di un ospedale
  - Questi dati sono protetti da leggi sulla privacy
  - Pertanto, questi dati non possono essere etichettati da agenzie esterne alla nostra organizzazione

# Problemi dell'etichettatura a mano: il tempo

- Altro problema da prendere in considerazione riguarda, invece, il **tempo necessario per etichettare tutti i dati**
- Per esempio, se vogliamo definire una trascrizione accurata di un enunciato vocale dobbiamo tener conto che:
  - In media, una trascrizione accurata (a livello fonetico) può impiegare un tempo 400 volte più grande della durata dell'enunciato
  - Quindi, se si vuole annotare un'ora di enunciato, ci vorranno circa 400 ore di etichettatura a mano
- Questa problematica va quindi ad influire sulla **capacità del modello di adattarsi ai cambiamenti** dell'ambiente in cui opera



## la molteplicità delle etichette

- Spesso, abbiamo necessità di moltissimi dati etichettati ed otteniamo questi dati da sorgenti diverse
  - **Sorgenti diverse** significa anche **annotatori diversi** e con expertise diversa.
- Questa diversità si traduce nella possibilità di avere **più interpretazioni di uno stesso concetto**
  - Potremmo trovarci in situazioni in cui uno stesso concetto viene etichettato con **etichette in completa contraddizione** tra loro.
- Questa problematica viene chiamata “**molteplicità delle etichette**” o “**label ambiguity**” ed è necessario che sia gestita al meglio.

## Un esempio di molteplicità delle etichette

- Consideriamo un task di entity recognition. Vogliamo etichettare diversi concetti in questa frase:

*Darth Sidious, known simply as the Emperor, was a Dark Lord of the Sith who reigned over the galaxy as Galactic Emperor of the First Galactic Empire*

- Affidiamo questo compito a 3 annotatori diversi e ci vengono restituiti 3 risultati diversi

Annotatore	N. Entità	Annotazione
1	3	[ <i>Darth Sidious</i> ], known simply as the Emperor, was a [ <i>Dark Lord of the Sith</i> ] who reigned over the galaxy as [ <i>Galactic Emperor of the First Galactic Empire</i> ].
2	6	[ <i>Darth Sidious</i> ], known simply as the [ <i>Emperor</i> ], was a [ <i>Dark Lord</i> ] of the [ <i>Sith</i> ] who reigned over the galaxy as [ <i>Galactic Emperor</i> ] of the [ <i>First Galactic Empire</i> ].
3	4	[ <i>Darth Sidious</i> ], known simply as the [ <i>Emperor</i> ], was a [ <i>Dark Lord of the Sith</i> ] who reigned over the galaxy as [ <i>Galactic Emperor of the First Galactic Empire</i> ].

## Un esempio di molteplicità delle etichette

- Questo tipo di situazione è molto comune:
  - La **diversa expertise** degli annotatori porta ad avere **disaccordi** sulle loro scelte
  - Più alto è il livello di expertise, maggiore sarà il rischio di annotazioni discordanti.
- Per ridurre questa problematica, è importante definire alcuni punti:
  - Innanzitutto, è fondamentale che gli annotatori abbiano una **chiara idea del problema** da trattare
  - Inoltre, è importante **definire un set di regole da seguire**, in modo tale che tutti gli annotatori abbiano **precise direttive**.

# Data Lineage

---

- Analizzare la **qualità** dei dati è fondamentale
  - Utilizzare dati provenienti da diverse sorgenti, ed etichettati da diversi annotatori, può portare il modello verso il fallimento e può farci sprecare un sacco di risorse
- Per esempio, abbiamo un modello che performa bene ma vogliamo migliorarlo
  - Lo addestriamo con un dataset più grande
  - Spendiamo molto tempo e denaro per raccogliere questi dati
  - Tuttavia, riaddestrando il modello, le prestazioni calano drasticamente

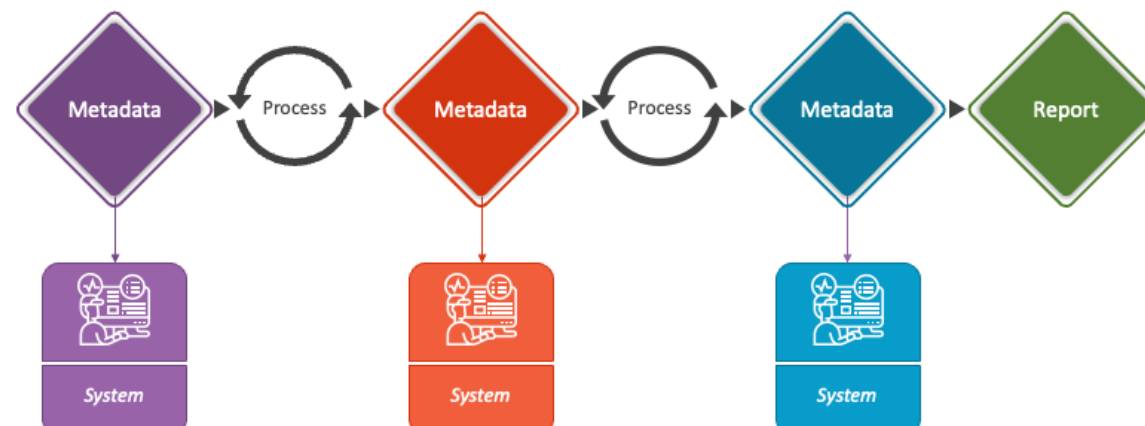
# Data lineage

---

- I motivi di questo degrado nelle prestazioni possono essere molteplici
- Uno di questi è strettamente legato al fatto che questi dati provengono da sorgenti diverse
  - Per esempio, gli annotatori sono stati meno attenti nell'etichettatura
  - C'è una differenza, in termini di accuratezza delle etichette, tra i nuovi dati ed i dati utilizzati in precedenza
- In questi casi, è importante tener traccia dell'origine dei dati che si utilizzano. Questa tecnica viene chiamata “Data lineage”.

# Data lineage

- Questa tecnica ha un duplice vantaggio:
  - Ci permette di individuare potenziali bias
  - Ci permette di fare debug del nostro modello facilmente
- Per esempio, se le performance del modello calano drasticamente con i nuovi dati, allora conviene analizzarli attentamente
  - Nella maggior parte dei casi, il problema non risiede nel modello ma nei dati utilizzati, a causa della presenza di molti errori di etichettatura.



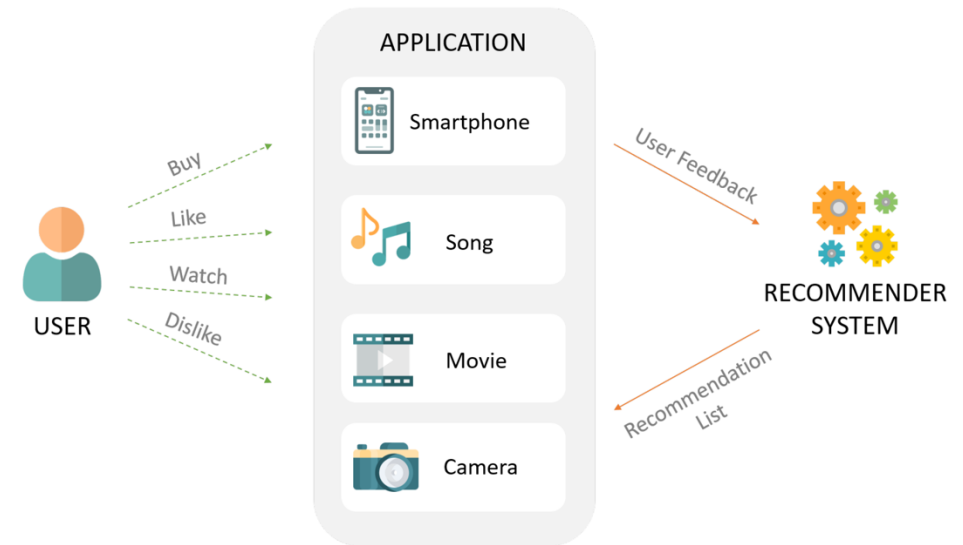
## Etichette naturali

---

- Come già detto in precedenza, esistono dati che non hanno bisogno di un processo di etichettatura a mano, in quanto **già hanno delle etichette assegnate**
  - Per esempio, Il modello alla base di Maps fa una stima del tempo necessario per arrivare ad una destinazione.
  - Quando la si raggiunge, il modello può confrontare la sua predizione con il tempo effettivo che è stato impiegato
- I task con etichette naturali, quindi, sono task dove le predizioni del modello possono **essere automaticamente valutate** dal sistema, senza l'intervento umano

# Un esempio di task con etichette naturali

- I sistemi di raccomandazione sono esempi perfetti per questi task
  - L'obiettivo di questi sistemi è raccomandare dei prodotti all'utente sulla base dei suoi acquisti/ricerche.
- Ogni volta che un utente clicca un prodotto raccomandato, il sistema può assumere che questo sia un esempio “positivo”
- Altri prodotti non cliccati possono essere visti come esempi “negativi”.





## Un esempio di task con etichette naturali

- In questo caso, si parla di “**etichette implicite**” ed “**etichette esplicite**”
  - Nell'esempio precedente, le etichette negative sono **implicite** perché la loro assegnazione deriva dalla mancanza di feedback diretti da parte dell'utente
  - Le etichette positive, invece, sono **esplicite** in quanto derivano da feedback (click).
- Ovviamente, presumere etichette implicite in maniera così semplice può essere molto dannoso per il nostro modello
  - L'utente potrebbe essere interessato al prodotto, ma per svariati motivi non ha avuto modo di visualizzarlo
  - Oppure, l'utente potrebbe aver premuto un prodotto per sbaglio

## Lunghezza del ciclo di feedback

- Di solito, quando si lavora con etichette naturali, il tempo diviene essenziale
  - Tornando all'esempio delle raccomandazioni, l'assegnazione di un'etichetta negativa può avvenire sulla base di un mancato click per un certo periodo di tempo
- **Lunghezza del ciclo di feedback:** il tempo che passa da quando una predizione viene fornita, fino a quando viene generato un feedback su di essa
- È fondamentale scegliere la giusta lunghezza del ciclo
  - Cicli brevi portano ad avere etichette in tempi brevi ma può portare ad errori
  - Con cicli lunghi potremmo dover aspettare settimane per ottenere delle etichette
- La scelta della giusta lunghezza implica un **compromesso accuratezza/velocità**

## Gestire la mancanza di etichette

- L'acquisizione di etichette di alta qualità è, ancora oggi, una sfida importante
- Esistono molte tecniche che trattano tale problematica. Le principali sono:

Metodo	Strategia	Base di partenza necessaria?
Weak supervision	Sfrutta euristiche (spesso rumorose) per generare etichette	<b>No</b> , ma un piccolo set di etichette è utile per guidare lo sviluppo delle euristiche
Semi-supervision	Sfrutta assunzioni strutturali per generare etichette	<b>Si</b> , un piccolo set di etichette è necessario come base di partenza per generare le altre
Transfer Learning	Sfrutta modelli pre-addestrati su altri task	- <b>No</b> , per zero-shot learning - <b>Si</b> , per il fine-tuning. Il set di base può essere anche molto piccolo
Active Learning	Etichetta i campioni di dati più importanti ed utili al tuo modello	<b>Si</b>

## Weak supervision

---

- Approccio divenuto molto popolare negli ultimi anni
- L'intuizione dietro questo approccio è legato all'uso di euristiche
  - Generalmente, le persone tendono a fidarsi delle euristiche
  - Queste vengono definite da persone esperte del dominio
- Per esempio, un dottore può usare delle euristiche per decidere se un paziente dovrebbe avere la priorità rispetto ad altri pazienti. L'euristica potrebbe essere:

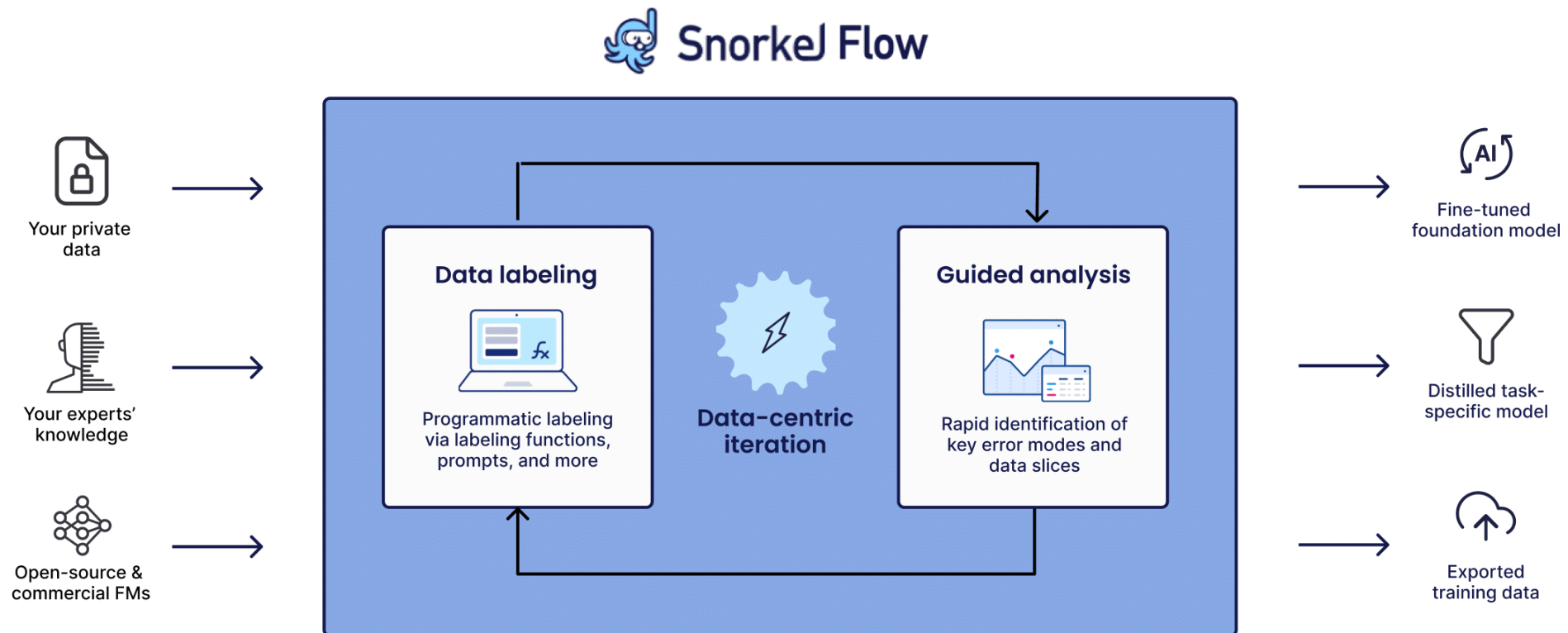
***“Se nelle note dell’infermiere vengono menzionate condizioni serie come una polmonite, allora questo paziente avrà una priorità maggiore rispetto ad altri”***

# La libreria Snorkel

- Una delle librerie più popolari per fare weak supervision è Snorkel

- Si basa sul concetto di “labeling function”
- Queste funzioni si occupano di codificare l'euristica

```
def labeling_function(note):  
    if "pneumonia" in note:  
        return "EMERGENT"
```



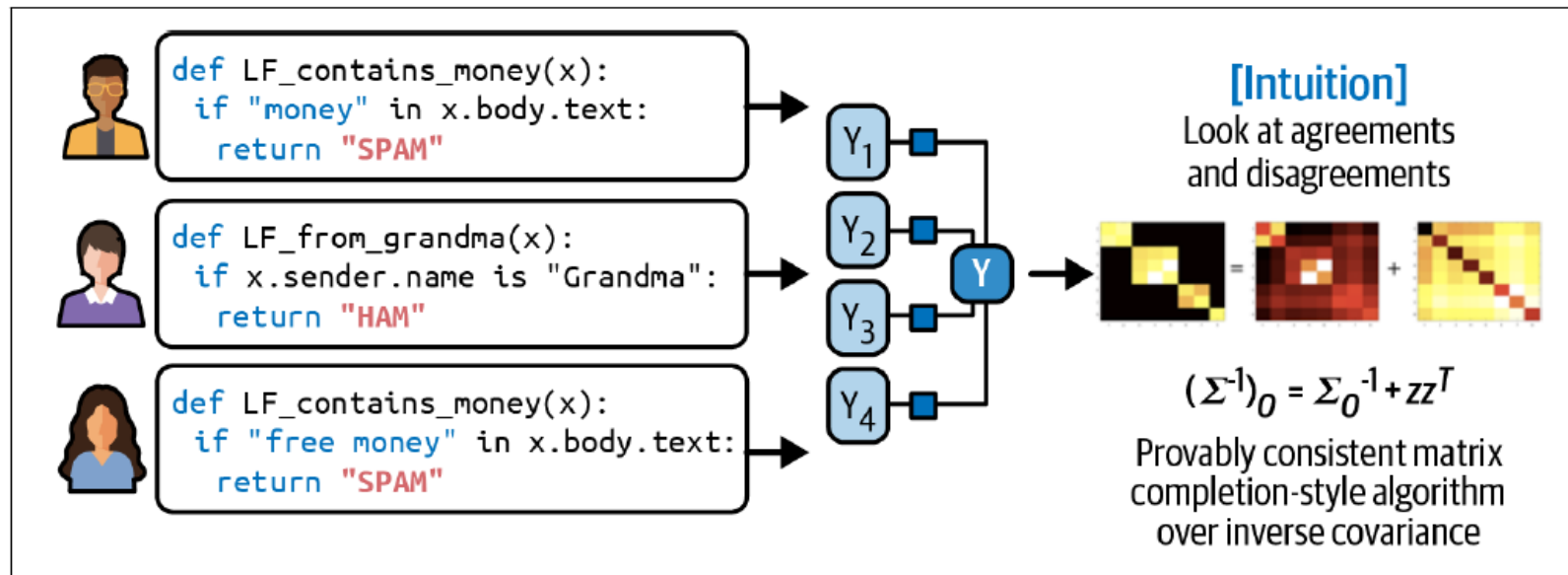
# Tipi di euristiche

---

- Esistono diverse tipologie di euristiche:
  - **Euristiche Keyword:** come quella vista nell'esempio precedente, si basa sull'individuazione di parole chiavi all'intero del campione
  - **Espressioni regolari:** queste sono basate sulla ricerca di un match di una certa espressione regolare definita
  - **Database lookup:** si definisce una lista di concetti. Il match avviene verificando se il campione contiene uno o più elementi della lista.
  - **Output di altri modelli:** le predizioni di altri modelli pre-addestrati diventano etichette

## Combinare le euristiche

- Spesso, le euristiche producono rumore e, di conseguenza, anche le etichette generate presentano rumore
  - È importante, quindi, combinare e ripesare le label function per ottenere un set di etichette corrette



# Etichettatura a mano vs etichettatura programmatica

Etichettatura a mano	Etichettatura programmatica
<b>Costosa:</b> specialmente quando è necessario l'intervento di annotatori esperti	<b>Cost saving:</b> expertise può essere condivisa e riusata
<b>Poca privacy:</b> necessità di rendere disponibili i dati ad annotatori	<b>Privacy:</b> creare LF usando sottocampioni dei dati. Queste LF possono essere usate per etichettare altri dati senza la necessità di esaminare singolarmente i campioni
<b>Lenta:</b> tempo richiesto scala linearmente con il numero di etichette necessarie	<b>Veloce:</b> scala molto facilmente
<b>Non adattiva:</b> ogni cambiamento comporta la necessità di rietichettare i dati	<b>Adattiva:</b> quando ci sono cambiamenti, ci basta riapplicare le LF

- La weak supervision, quindi, è un approccio **semplice e potente**.
  - Ci offre un **buon punto di partenza** per l'etichettatura dei dati
  - Ma bisogna **stare attenti al rumore generato** dalle euristiche che definiamo



# Semi-supervision

---

- L'approccio weak-supervision può portare a generare rumore nelle etichette
- Una soluzione a tale problematica viene data dall'approccio **Semi-supervision**
  - Si definiscono delle assunzioni strutturali per generare nuove etichette
  - Questo approccio necessita di un set iniziale di etichette
- Le prime tecniche semi-supervision sono state sviluppate negli anni 90'.

Esistono diverse tecniche di questo tipo, tra cui:

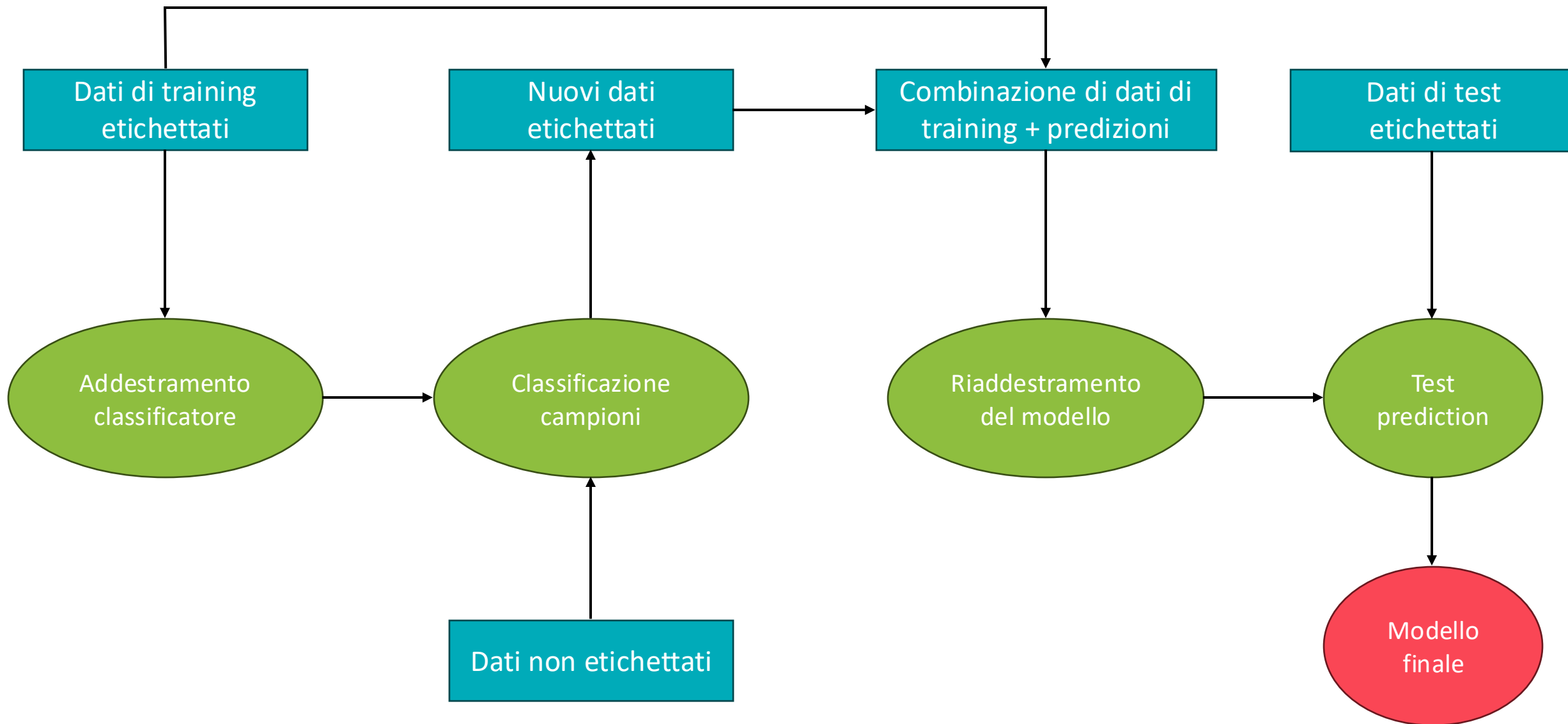
- **Self-Training**
- **Similarity-based**
- **Perturbation-based**

# Self-training

---

- La più classica tecnica semi-supervision
  - Inizialmente, **si addestra un modello** sul set di dati etichettati disponibile
  - Si usa questo modello per **fare predizioni dei dati non etichettati**
- Assumendo che le predizioni con alta probabilità siano corrette, possiamo **aggiungere** queste etichette al set già disponibile.
  - In questo modo, quindi, il dataset di training diviene via via **più grande** e si può sfruttare per riaddestrare il modello
  - Questa operazione viene ripetuta fino a quando non si ottengono modelli con performance accettabili

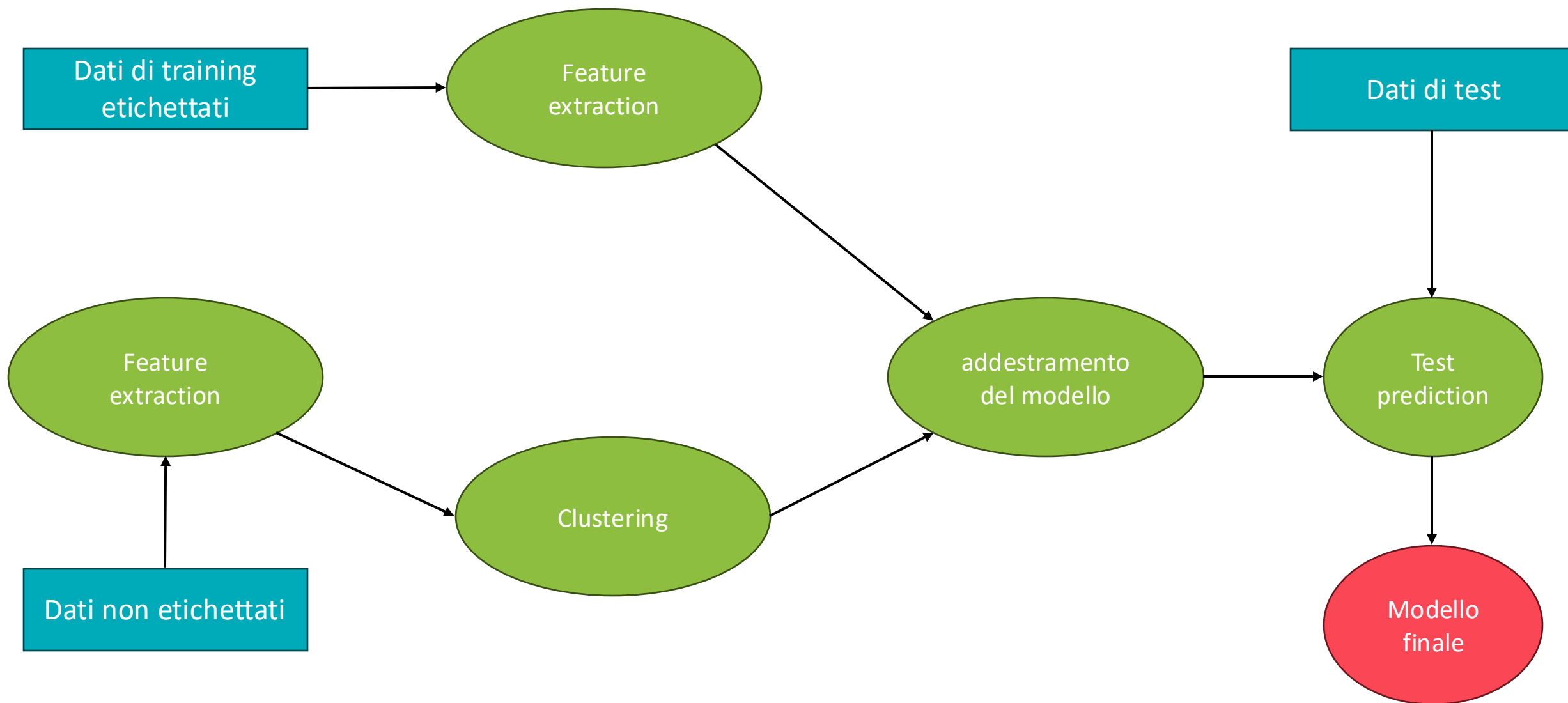
# Self-training



## Metodo Similarity-based

- La seconda tecnica semi-supervision, invece, lavora sulla similarità dei campioni
  - Si assume che dati con **caratteristiche simili**, debbano condividere la stessa etichetta
- La similarità può essere valutata con metodi semplici o con metodi complessi
- Per esempio, vogliamo assegnare un topic ad una serie di hashtag
  - Possiamo assumere che tutti gli hashtag presenti in uno stesso tweet fanno riferimento allo stesso topic **(similarità semplice)**
  - Oppure, possiamo utilizzare il clustering per raggruppare gli hashtag in maniera tale che hashtag simili appartengano allo stesso gruppo **(Similarità complessa)**

## Metodo Similarity-based



## Metodo Perturbation-based

- Una tecnica molto recente viene definita come “**Perturbation-based**”
  - Con questa tecnica, si va ad **aggiungere volontariamente** del rumore ai dati
- L'intuizione alla base di questa tecnica è che **con piccoli cambiamenti** (perturbazioni), le etichette **non dovrebbero cambiare**
- Le perturbazioni possono essere di diverso tipo:
  - Si può aggiungere, per esempio, del rumore alle immagini
  - Si possono aggiungere piccoli valori generati randomicamente
  - Si possono modificare, di poco, gli embedding di una serie di parole
- È importante, però, che i dati perturbati **abbiamo le stesse etichette** di quelli non perturbati.

## Le tecniche semi-supervision funzionano?

- In genere, queste tecniche performano in **maniera ottimale** quando si hanno **poche etichette per l'addestramento** dei modelli
- Un aspetto da non sottovalutare, però, riguarda l'ammontare di dati da utilizzare in fase di valutazione del modello
  - Se ne utilizziamo pochi, il modello potrebbe soffrire di **overfitting**
  - Se ne utilizziamo troppi, l'aumento delle prestazioni ottenute scegliendo il modello migliore **potrebbe essere inferiore** all'aumento che si otterrebbe utilizzando questi dati per la fase di training.
- È fondamentale, quindi, tenere in considerazione questo **compromesso**

# Transfer learning

---

- Con il termine “**transfer learning**” si fa riferimento alla famiglia di metodi dove un modello, sviluppato per un task, **viene riusato** come **punto di partenza** per un modello che lavora per un altro task.
  - Inizialmente, il modello base è addestrato per un task base (spesso economico e abbondante di dati di training)
  - Questo modello addestrato, allora, può essere utilizzato per il task di nostro interesse
- In alcune situazioni, noi possiamo utilizzare direttamente il modello base per performare il task (**zero-shot learning**)
- In altri casi, invece, il modello deve essere raffinato per essere utilizzato con il nuovo task (**fine-tuning**)



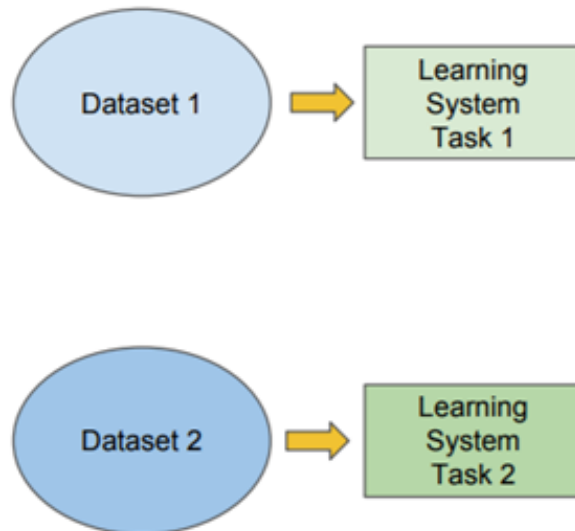
# Transfer learning\*\*

## Traditional ML

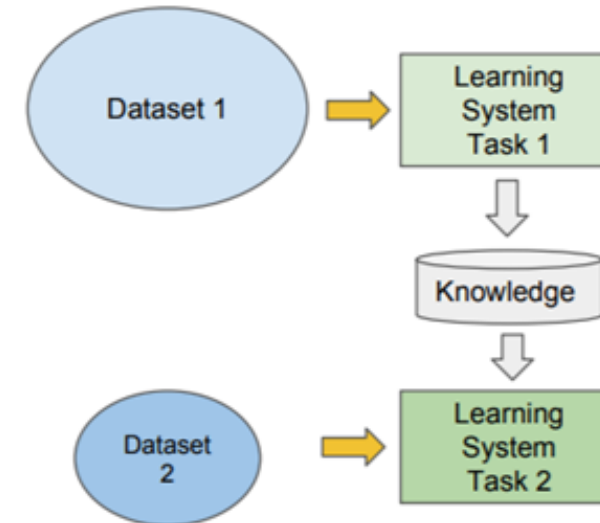
vs

## Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

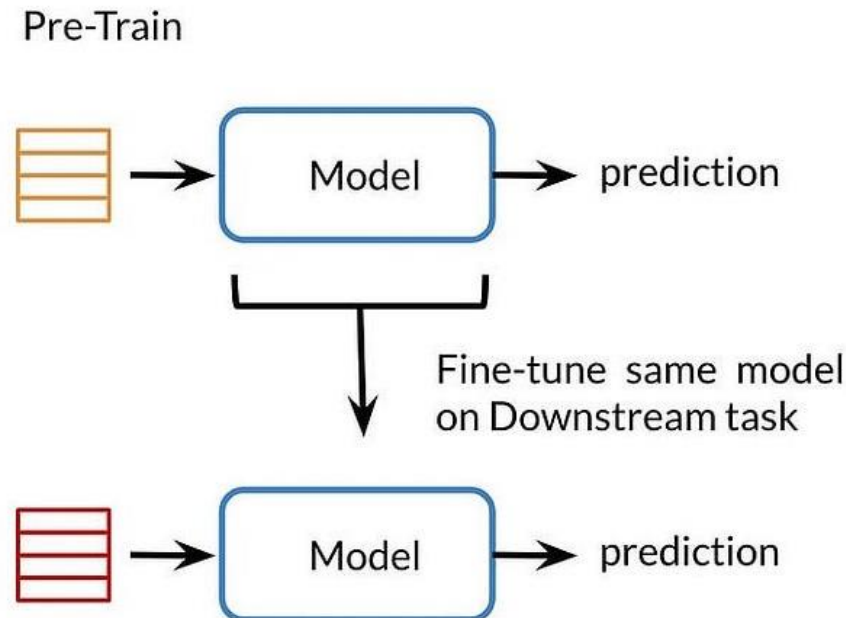


- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data

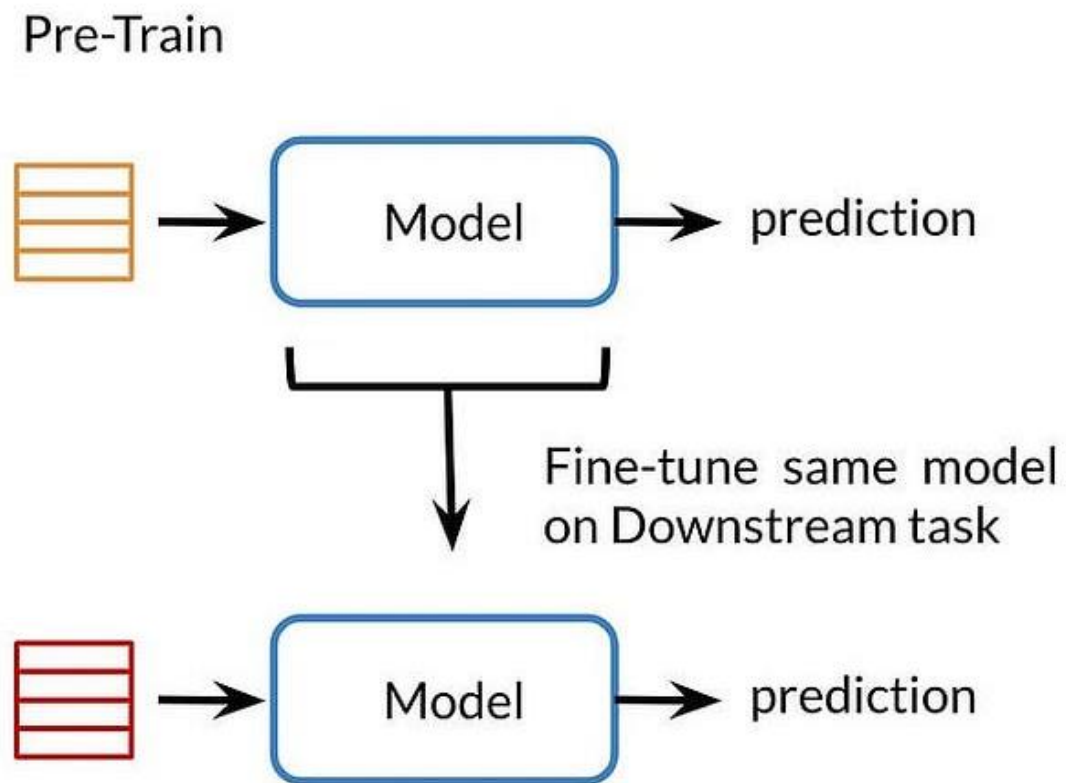
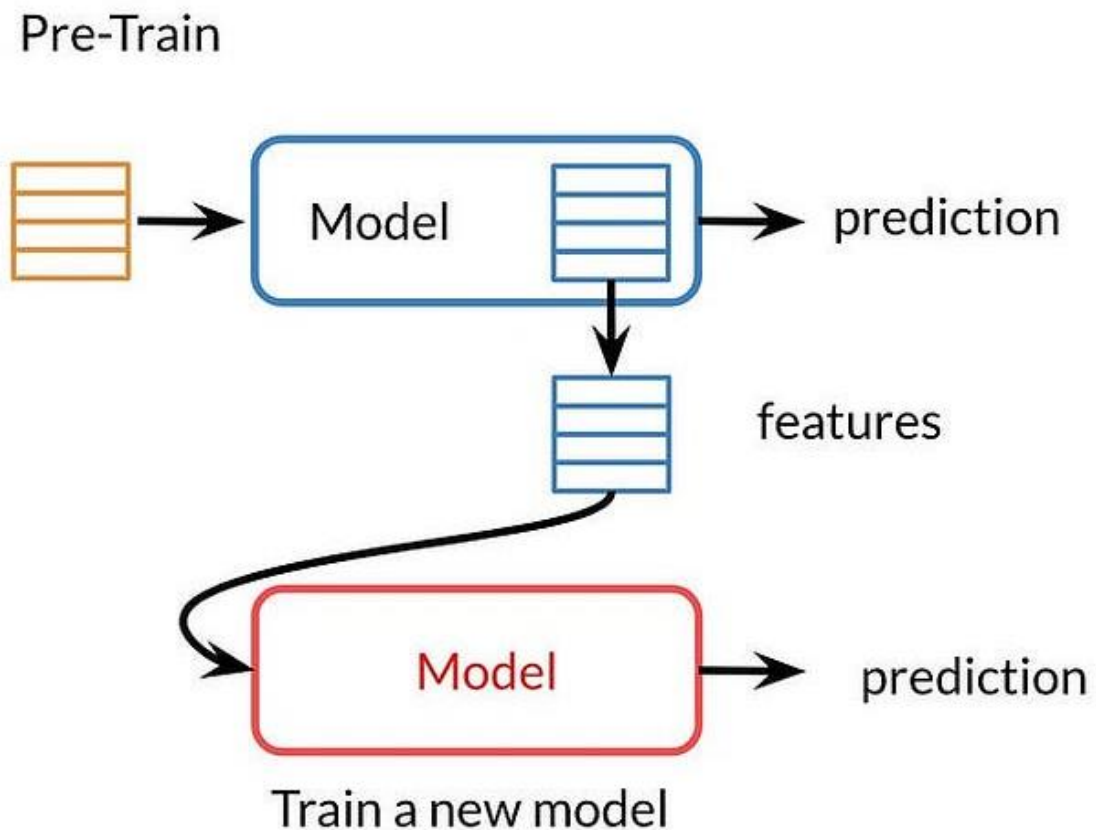


# Fine tuning di un modello

- Fare fine-tuning di un modello significa apportare delle piccole modifiche al modello base
  - In genere, si continua l'addestramento del modello base, o di una parte di esso, con un set di dati di training relativi al nuovo task che si vuole affrontare



# Feature-based vs Fine-Tuning



# Potenzialità del transfer learning

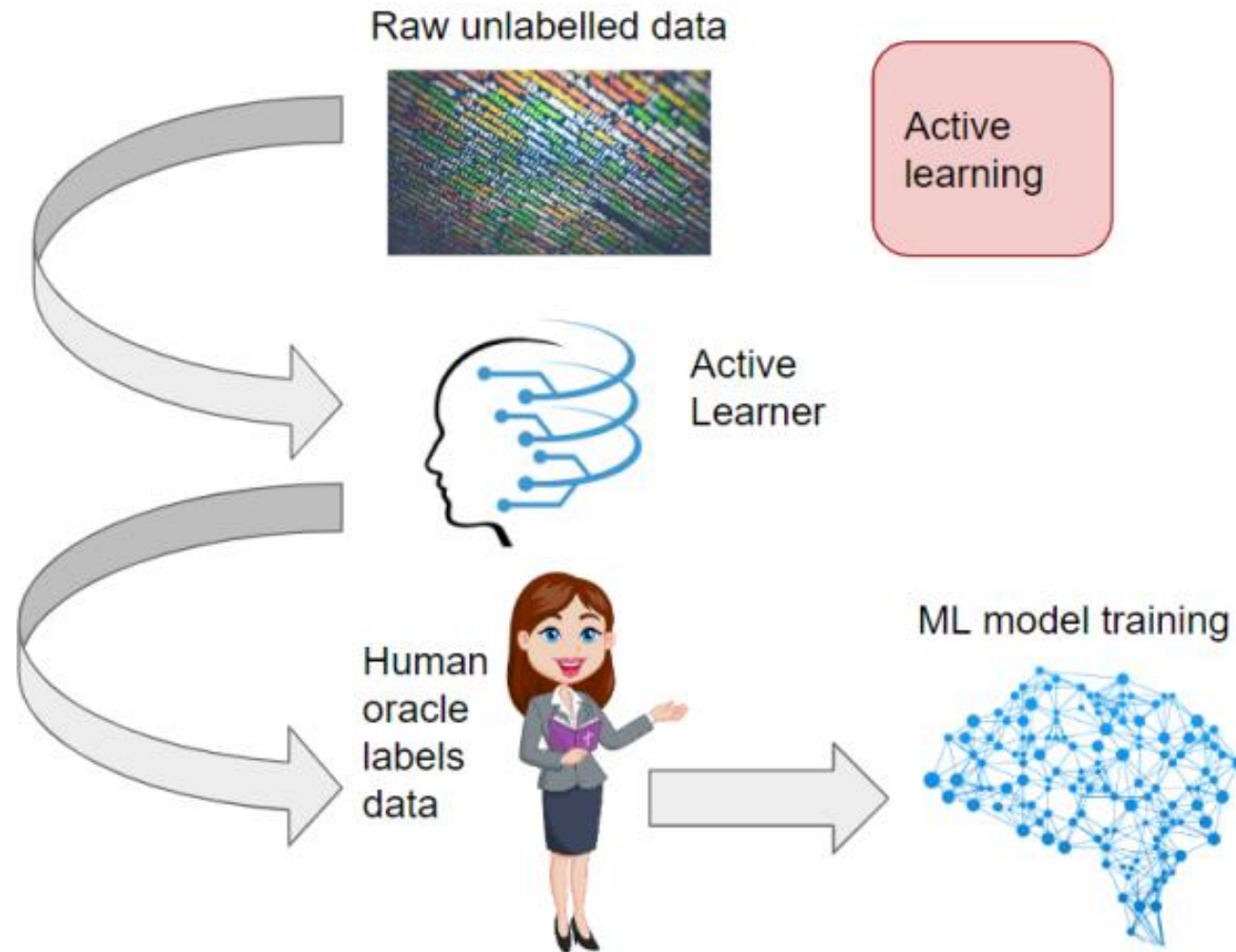
- Il transfer learning è molto utile quando **non si hanno molti dati** etichettati
  - Tuttavia, risulta essere utile anche quando abbiamo dati etichettati e vogliamo **migliorare le performance di un modello sfruttando modelli molto performanti**
- Negli ultimi anni, questa tecnica ha permesso lo sviluppo di modelli ad alte prestazioni per task molto complessi
  - Pochi dati disponibili o enormi quantità di risorse necessarie
- Modelli come GPT e BERT sono il frutto dell'applicazione di tale tecnica.
- In futuro, questo tipo di tecnica verrà utilizzata sempre più spesso ed in maniera sempre più determinante

# Active learning

---

- L'ultima tecnica che vedremo è chiamata “**active learning**” e si pone l'obiettivo di migliorare l'efficienza dei dati etichettati
- In questo caso, l'idea è che i modelli di ML possono ottenere prestazioni migliori con pochi dati etichettati se questi modelli hanno **la possibilità di scegliere su quali dati addestrarsi**
- Viene spesso chiamata “**query learning**” in quanto:
  - Un modello (active learner) invia delle query (dati non etichettati) ad annotatori (solitamente, umani) che hanno il compito di etichettarli.

# Active learning



## Misura di incertezza

---

- Con questa tecnica, i dati non vengono più etichettati in maniera randomica
  - Piuttosto, gli annotatori etichettano solo quelli che ritengono più utili per l'addestramento del modello
  - L'utilità viene valutata attraverso l'uso di metriche ed euristiche
- La metrica più utile per tale scopo è la “misura di incertezza”
  - Si etichettano i campioni su cui il modello è più indeciso
  - In questo modo, si presuppone che questi dati possano aiutare il modello a definire meglio il decision boundary

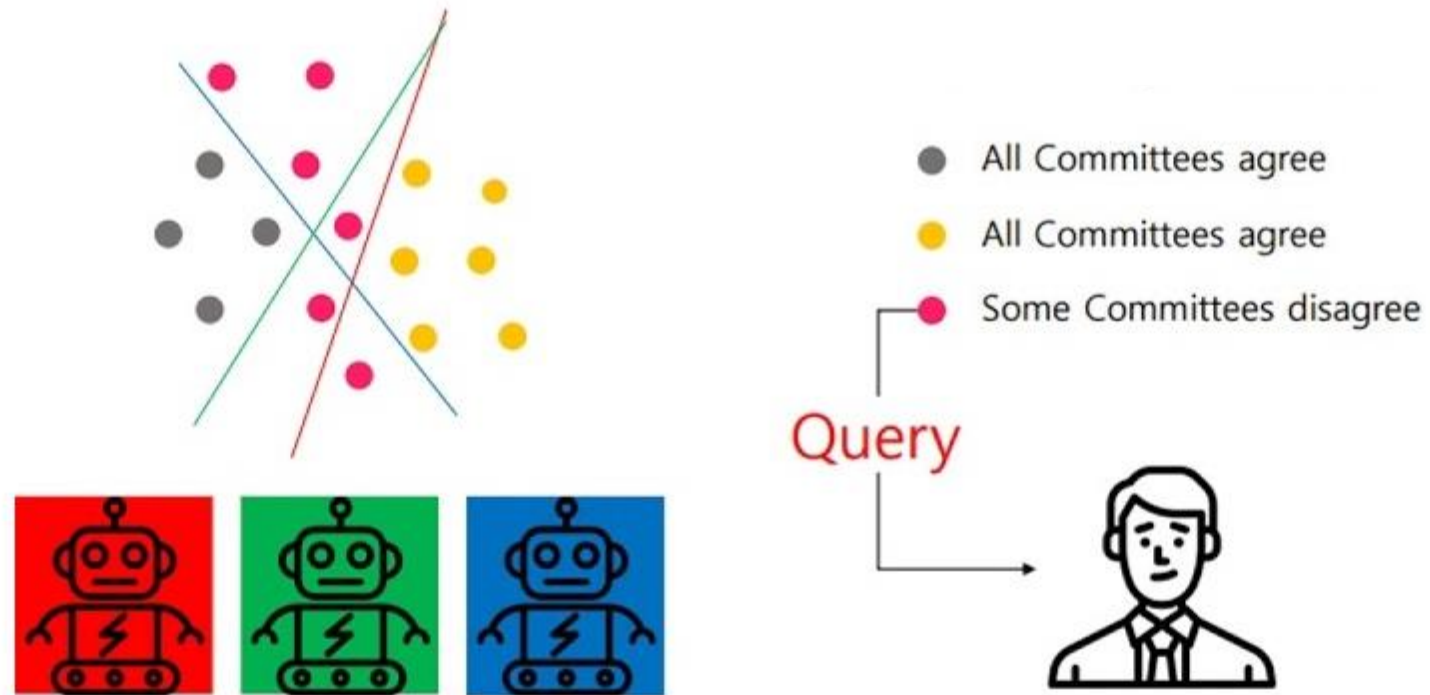
## Query-by-committee

---

- Un'altra euristica molto utilizzata è basata sul “disaccordo tra più modelli candidati” o “Query-by-committee”
  - In questo caso, abbiamo un “comitato” composto da diversi modelli (in genere, è lo stesso modello ma addestrato con parti di dataset e configurazioni diverse)
  - Ognuno di questi modelli decide quale campione etichettare (basandosi sull'incertezza della sua predizione)
  - Una volta che abbiamo tutte le predizioni, calcoliamo l'incertezza e si scelgono i campioni più incerti.



# Query-by-committee



## Potenzialità del Active Learning

- L'active learning sta diventando sempre più popolare grazie alla sua **utilità nel mondo dei data stream e dei dati real-time**
  - In questi contesti, I dati tendono a **cambiare molto e molto rapidamente**
- È fondamentale, quindi, avere modelli che si **adattano facilmente e velocemente** a questi cambiamenti
- L'active learning rappresenta un'ottima strategia per la gestione di tale fenomeno e ci permette di ottenere **modelli real-time ad alta efficienza.**

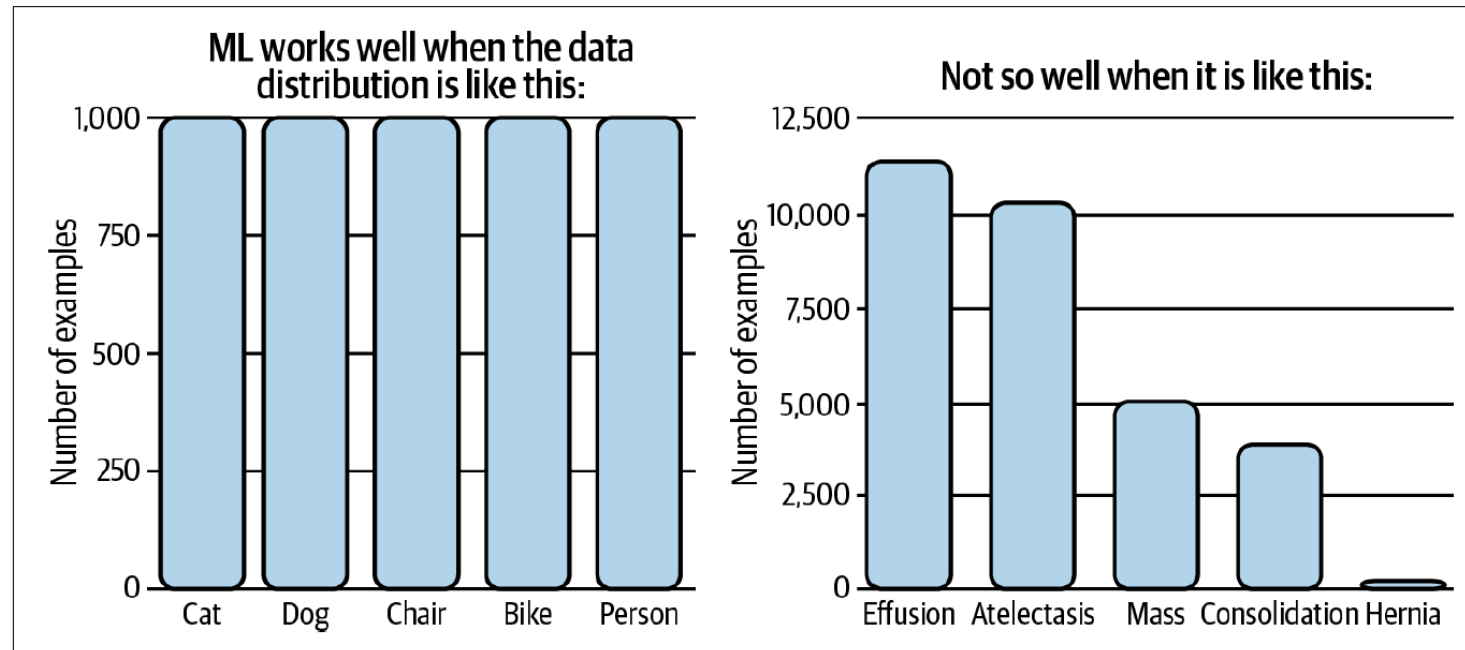
# Bilanciamento delle classi

# Un problema molto frequente

- Quando si lavora con modelli di ML, uno dei problemi più comuni è legato allo “**sbilanciamento delle classi**”
- Si parla di “**classi sbilanciate**” quando c’è una **grande differenza**, in termini di **numero di campioni**, tra le classi dei dati di training
  - Per esempio, in un dataset di immagini X-ray, abbiamo il 90% di immagini di persone sane e solo il 10% di immagini di persone malate.
- Questo problema incide moltissimo quando lavoriamo su task di classificazione, ma anche task di regressione possono soffrirne

# Classi sbilanciate: problematiche

- I modelli di ML lavorano bene in situazioni dove la distribuzione dei dati è più bilanciata



- Classi sbilanciate rendono difficile l'apprendimento per diversi motivi

# Classi sbilanciate: problematiche

- Una prima motivazione è legata al fatto che il nostro modello potrebbe **non avere campioni sufficienti per riuscire a classificare** le classi minoritarie
  - Se abbiamo poche istanze di una determinata classe, il problema diventa un “few-shot learning”, dove il nostro modello vede poche volte una determinata classe e non riesce a prendere una decisione su di essa
  - Se, invece, non si hanno istanze di una determinata classe, il modello assumerà che quella classe non esiste.

# Classi sbilanciate: problematiche

- Un'altra problematica riguarda l'ottimalità del modello
  - Con dataset sbilanciati, un modello potrebbe rimanere bloccato in una **soluzione non ottimale**
  - Potrebbe, quindi, **non apprendere pattern e strutture** presenti nei dati
- Considerando un modello di classificazione per le immagini X-ray
  - Utilizzando un dataset come quello descritto in precedenza (90% sani, 10% malati), il modello restituirà quasi sempre la classe maggioritaria con un'accuratezza molto elevata.
  - Aggiungendo, però, randomicità nei campioni, le prestazioni del modello inizierebbero a degradarsi moltissimo

# Gestire dataset sbilanciati

- Esistono diverse strategie per la gestione di questa problematica:
  - **Strategie metric-level**
  - **Strategie data-level**
  - **Strategie algorithm-level**
- Questi metodi possono mitigare di molto gli effetti dello sbilanciamento
  - Tuttavia, possono non essere sufficienti alla risoluzione del problema
- Sviluppare un modello abbastanza buono può essere molto complicato anche se si sfruttano tali tecniche



# Tecniche metric-level: scegliere le giuste metriche

- La prima strategia di mitigazione si basa sulla **scelta di metriche** appropriate per la gestione e la valutazione dello sbilanciamento dei dati
  - La scelta delle metriche è **uno dei passi fondamentali** quando si ha a che fare con questo tipo di dataset
- Utilizzare metriche errate porta ad avere un'idea errata riguardo alle prestazioni del modello
  - Di conseguenza, diviene molto complicato prevedere dei miglioramenti al modello

# Tecniche metric-level: scegliere le giuste metriche

- L'accuracy e l'error rate sono le metriche più comunemente utilizzate
- Tuttavia, queste metriche **non sono adatte** in situazioni di dataset sbilanciati
  - Esse trattano tutte le classi **equamente**
  - La classe maggioritaria, quindi, **dominerà queste metriche** condizionandole in maniera sostanziale
- Questa problematica diviene ancora più determinante quando la classe maggioritaria non è quella su cui bisogna porre maggior attenzione
  - Nell'esempio precedente, classificare correttamente immagini di persone sane è meno importante di classificare correttamente immagini di persone malate

# Tecniche metrics-level: scegliere le giuste metriche

- Riprendendo l'esempio proposto in precedenza
- Supponiamo di avere 1000 immagini, di cui
  - 100 sono immagini etichettate come “cancro”
  - 900 sono immagini etichettate come “normali”
- Consideriamo due modelli di classificazione A e B con i seguenti risultati:

Modello A	Vere Cancro	Vere Normali
Predette Cancro	10	10
Predette Normali	90	890

Modello B	Vere Cancro	Vere Normali
Predette Cancro	90	90
Predette Normali	10	810

## Tecniche metrics-level: scegliere le giuste metriche

Modello A	Vere Cancro	Vere Normali
Predette Cancro	10	10
Predette Normali	90	890

Modello B	Vere Cancro	Vere Normali
Predette Cancro	90	90
Predette Normali	10	810

- Ad una prima analisi, il modello B sembrerebbe quello ideale, in quanto ha probabilità maggiori di classificare correttamente le immagini con cancro
  - Tuttavia, però, entrambi i modelli hanno un accuracy pari a 0.9
- L'accuracy non ci fornisce una precisa indicazione su quale modello sia meglio
  - Possiamo utilizzarla lo stesso, ma dovrebbe essere calcolata per ogni classe individualmente.

# Precision, Recall e F1-Score

- In una semplice classificazione binaria, possiamo calcolare le performance del modello rispetto ad una classe considerata “positiva”

	Prediz. positive	Prediz. negative
Label positive	Veri positivi (TP)	Falsi negativi (FN)
Label negative	Falsi positivi (FP)	Veri negativi (TN)

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

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

$$F1 \text{ Score} = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

- Queste metriche sono **asimmetriche**
  - Il loro valore dipende da quale classe viene considerata come positiva

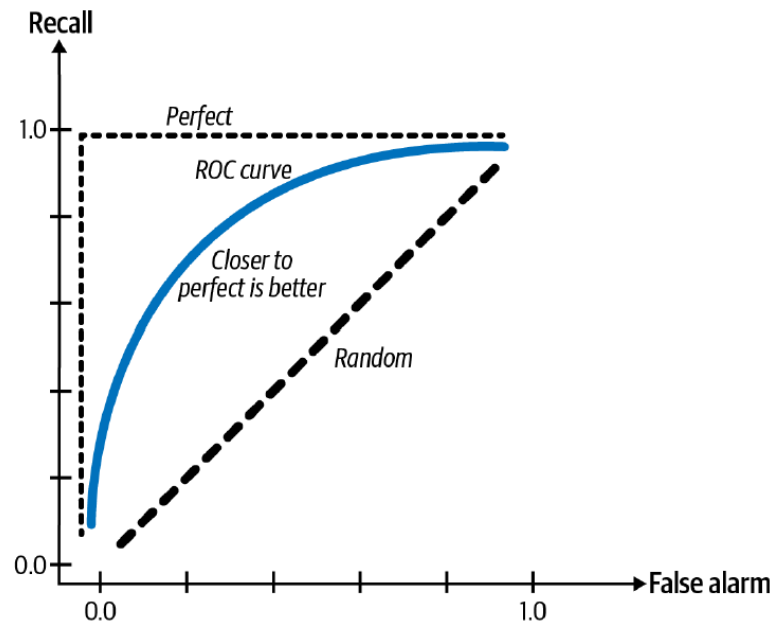
# ROC Curve

---

- Molti task di classificazione possono essere modellati come task di regressione
  - Il modello può restituire una probabilità sulla quale classifichiamo i dati
- Possiamo definire una soglia che ci permette di classificare un campione come positivo o negativo
  - In questo modo, noi possiamo influenzare le metriche che calcoliamo
  - Possiamo, ad esempio, incrementare la recall decrementando i falsi positivi
- Possiamo raffigurare l'effetto di differenti threshold attraverso un plot chiamato **ROC Curve**
  - Come per le altre metriche, la ROC Curve si concentra solo sulla classe positiva

# ROC Curve

- Con questo plot, possiamo osservare l'effetto di differenti threshold sulle performance del modello
- Quando la Recall è pari a 1 (100%), il modello è **perfetto**
- Possiamo calcolare **l'area sotto la curva (AUC)**. Più ci avviciniamo alla perfezione, maggiore sarà la grandezza di quest'area



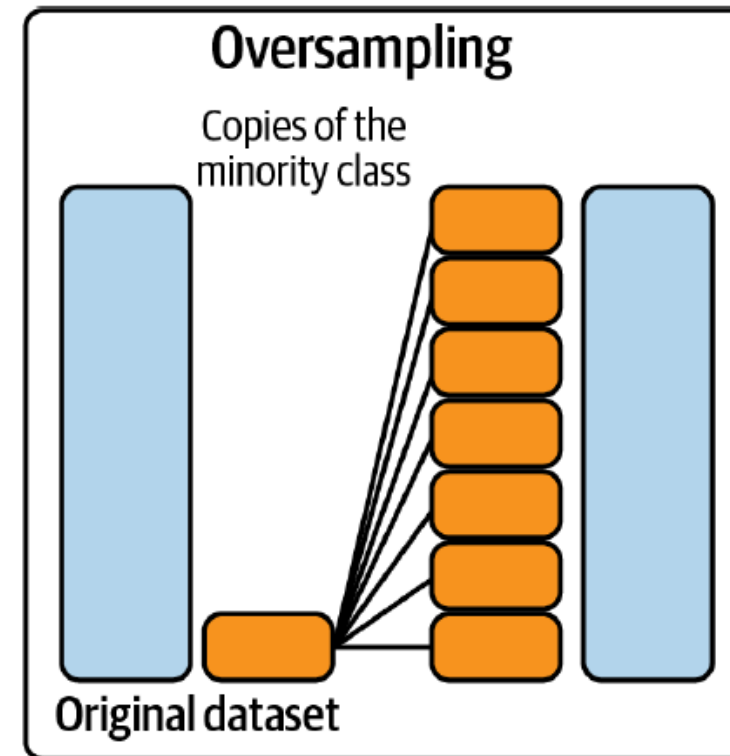
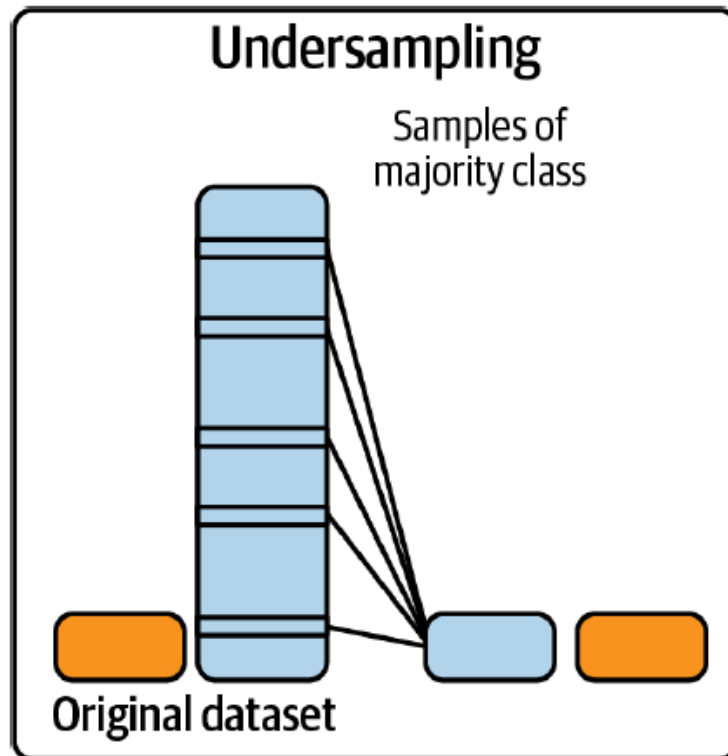
# Tecniche data-level

- Oltre alla scelta delle giuste metriche, esistono altri metodi che intervengono direttamente sui dati
  - Queste tecniche vanno a **modificare direttamente la distribuzione dei dati** di training in modo tale da ridurre il livello di sbilanciamento, favorendo l'apprendimento del modello
- Tra queste tecniche, le più comuni sono le tecniche di **Ricampionamento**.  
Tra queste, le più importanti sono:
  - **Sottocampionamento**
  - **Sovracampionamento**
  - **Apprendimento a due fasi**
  - **Campionamento dinamico**



# Ricampionamento

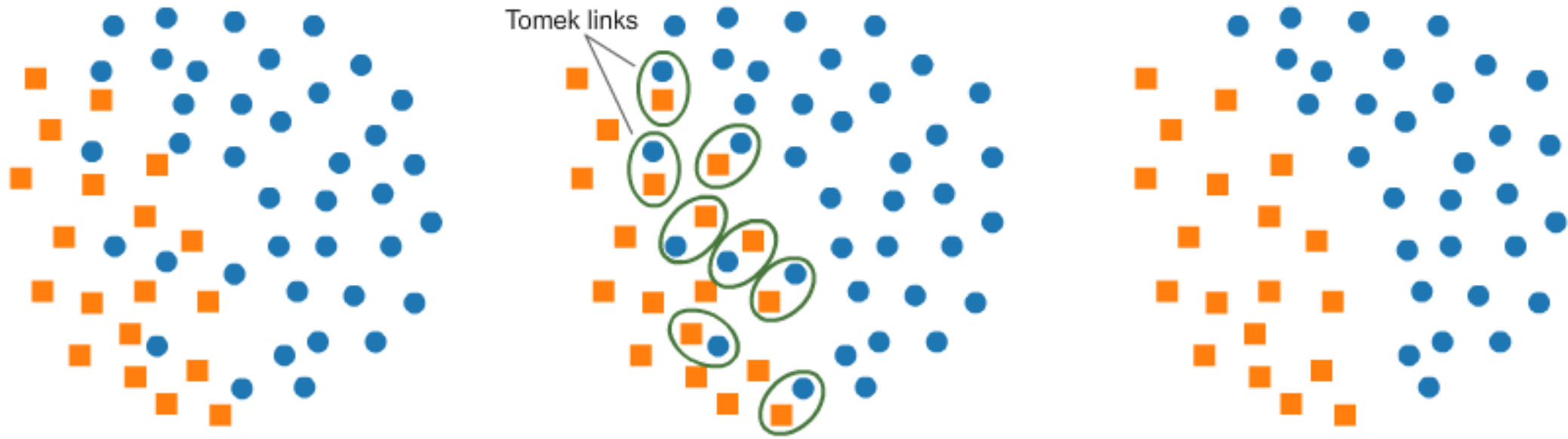
- Quando si parla di ricampionamento, si fa riferimento a tutte quelle tecniche che vanno ad **aggiungere o rimuovere** campioni dal dataset per bilanciare lo stesso



# Sottocampionamento

- La tecnica del sottocampionamento è concettualmente molto semplice
  - Si vanno a rimuovere campioni della classe maggioritaria in modo tale da ridurre la differenza con quella minoritaria
- Il modo più semplice per fare ciò è rimuovere randomicamente i campioni
- Tuttavia, esistono altre tecniche di questo tipo, come la tecnica **Tomek**
  - Metodo popolare per il trattamento di dati a bassa dimensionalità
  - Si generano coppie di campioni di classi opposte sulla base della loro vicinanza
  - Si rimuove il campione della classe maggioritaria per ogni coppia

# Sottocampionamento Tomek link

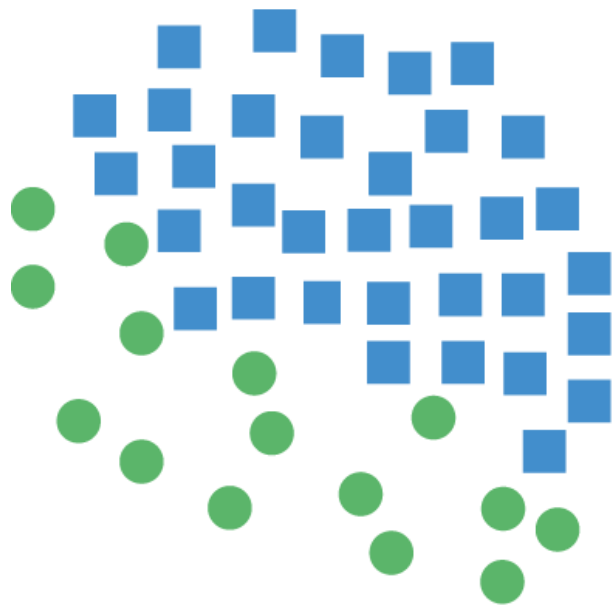


# Sovracampionamento

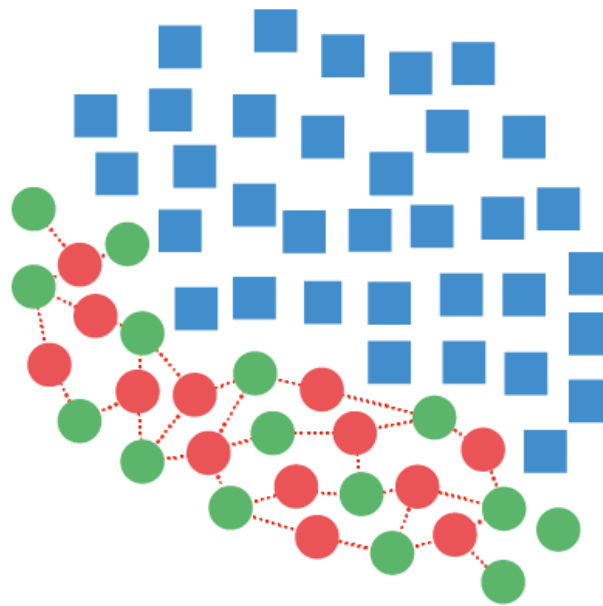
- Il sovracampionamento, invece, lavora in maniera opposta
  - Si vanno ad aggiungere campioni della classe minoritaria in modo tale da ridurre la differenza con quella maggioritaria
- Si può sovracampionare un dataset in maniera molto semplice, andando a generare copie di campioni minoritari in maniera randomica
- Anche in questo caso, esistono altre tecniche come la tecnica **SMOTE**
  - Anch'essa lavora bene con dati a bassa dimensionalità
  - Si sintetizzano nuovi campioni della classe minoritaria
  - Sintesi che avviene campionando combinazioni convesse di data point esistenti all'interno della classe minoritaria

# Sottocampionamento SMOTE

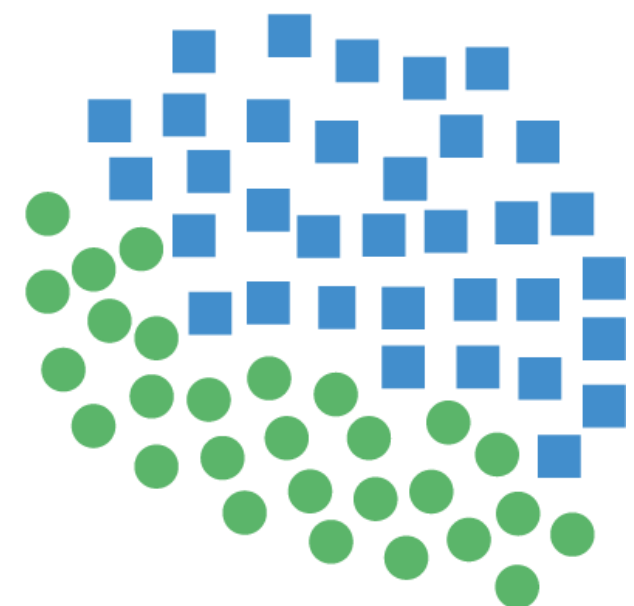
## Synthetic Minority Oversampling Technique



Original Dataset



Generating Samples



Resampled Dataset

# Apprendimento a due fasi

- Sovracampionamento e Sottocampionamento hanno aspetti critici
  - Con il primo, si rischia di avere overfitting del modello
  - Con il secondo, si rischia di perdere dati importanti
- Per mitigare questi effetti, sono state definite tecniche più sofisticate, tra cui l'apprendimento a due fasi
  - Si sottocampiona il dataset fino ad avere  $N$  campioni per ogni classe
  - Successivamente, si addestra il modello con il dataset ricampionato
  - Infine, si fa fine-tuning del modello utilizzando il dataset originale

# Campionamento dinamico

- Altra tecnica che cerca di superare i limiti di sottocampionamento e sovracampionamento
  - Viene applicata durante il processo di addestramento del modello
  - Si sovracampionano le classi meno performanti
  - E si sottocampionano quelle più performanti
- Questo metodo ci permette di migliorare il modello in quanto:
  - Gli viene mostrata meno la classe che ha già imparato a predire
  - Conseguentemente, gli vengono mostrati più campioni della classe su cui ha ancora difficoltà

# Tecniche algorithm-level

- L'ultima famiglia di tecniche che vedremo è chiamata “**algorithm-level**”
  - A differenza delle tecniche viste in precedenza, non vanno a lavorare direttamente sui dati di training, ma **apportano delle modifiche a livello di algoritmo d'apprendimento** per mitigare l'effetto dello sbilanciamento
- Rendono il modello più robusto allo sbilanciamento del dataset
- Tra queste tecniche, le principali sono:
  - **Cost-sensitive learning**
  - **Class-balanced loss**
  - **Focal loss**



# Tecniche algorithm-level

- In generale, queste tecniche lavorano sulla **loss function**
  - Essendo la funzione che guida l'apprendimento del modello, si tenta di apportare aggiustamenti ad essa per migliorare le prestazioni
- **Idea**: dare priorità alle predizioni che producono una perdita maggiore
- Supponiamo di avere due campioni,  $x$  e  $y$ , per cui vengono fatte predizioni errate:
  - la loss ottenuta dalla predizione di  $x$  è maggiore di quella ottenuta con  $y$
  - in questo caso, fare predizioni corrette per  $x$  ha una priorità più alta rispetto a  $y$
  - Assegnando dei **pesi maggiori** ai campioni con **maggiore priorità**, spingiamo il modello a **concentrarsi maggiormente** su di essi

# Cost-sensitive learning

- Nata nel 2001 da un'intuizione di Charles Elkan
  - L'errata classificazione di differenti classi produce differenti costi
- La loss function viene modificata in modo tale da prendere in considerazione questa variabilità dei costi
  - Si inizia con una matrice di costo per definire un costo  $C_{ij}$
  - Costo  $C_{ij}$  = costo di classificare la classe  $i$  con la classe  $j$  ( $i = j$  allora costo = 0)
  - La loss ottenuta da un'istanza  $x$  di una classe  $i$  verrà calcolata come la media pesata di tutte le possibili classificazioni dell'istanza  $x$

$$L(x; \theta) = \sum_j C_{ij} P(j|x; \theta)$$

- **Problema:** necessità di definire manualmente la matrice

# Class-balanced loss

- Tecnica che prevede una “punizione” per il modello quando produce previsioni errate sulla classe minoritaria
- Versione base della tecnica:
  - Assegniamo un peso ad ogni classe in maniera inversamente proporzionale al loro numero di campioni
  - Classi minoritarie o rare avranno un peso maggiore rispetto alle altre
  - La funzione di loss diventa la seguente:

$$L(x; \theta) = W_i \sum_j P(j|x; \theta) \text{Loss}(x, j)$$

- Versioni più sofisticate di questa tecnica possono prendere in considerazione la sovrapposizione tra campioni esistenti

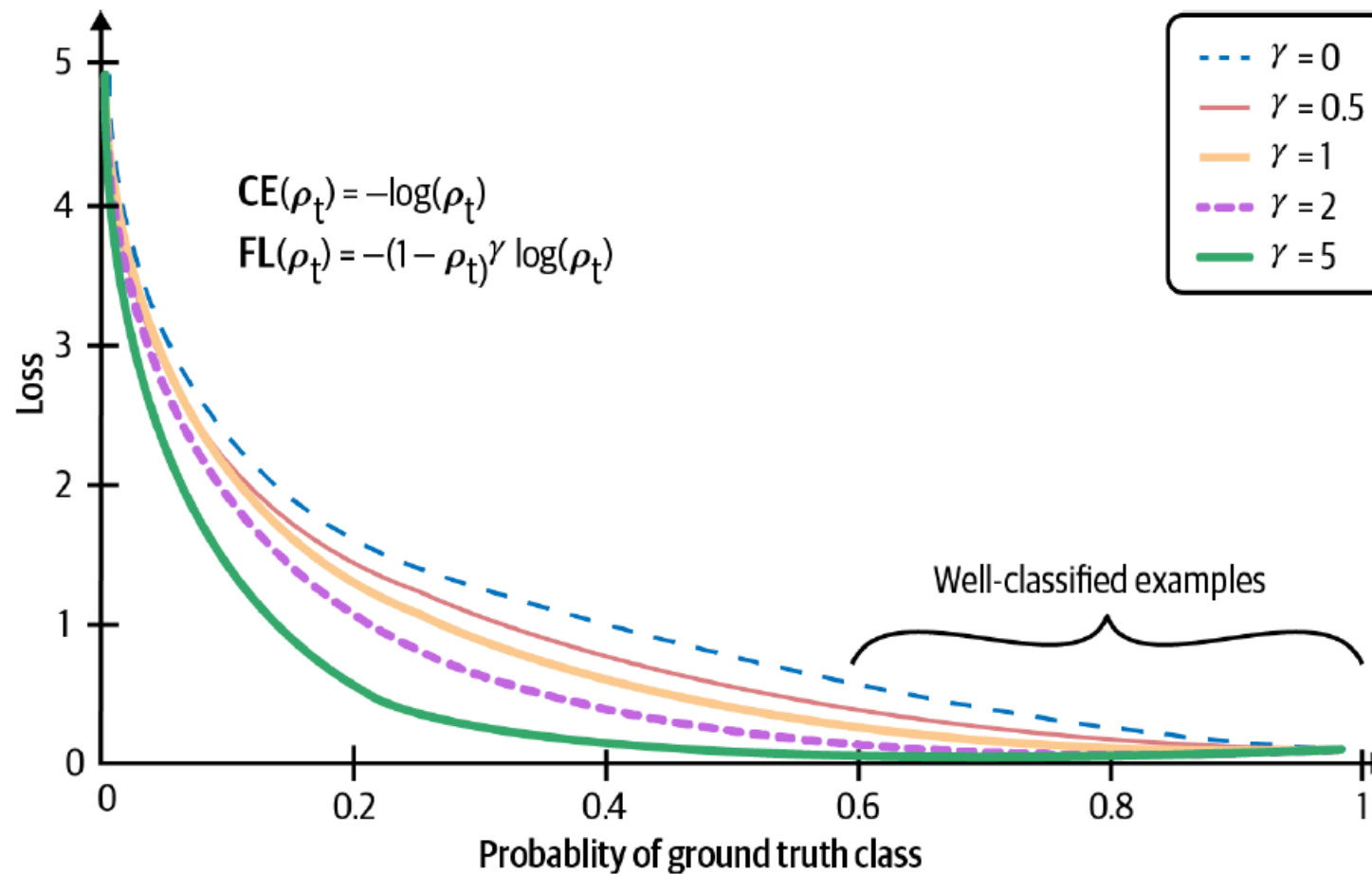
# Focal loss

---

- L'ultima tecnica che analizziamo è la “Focal loss”
  - Nei dati, è possibile avere campioni più semplici da classificare rispetto ad altri
  - Per questi campioni, il modello impara molto velocemente
  - Si vuole incentivare il modello a concentrarsi maggiormente sui campioni più difficili (con una probabilità di correttezza di predizione più bassa)
- **Idea:** regolare la loss in modo tale assegnare un peso Maggiore a quei campioni su cui il modello ha maggiori difficoltà

# Focal loss

- In questo grafico, viene illustrata una comparazione tra la Cross Entropy function e la Focal loss function



# Gestire lo sbilanciamento è ancora una sfida aperta

- Abbiamo visto che per gestire questa problematica esistono molte tecniche differenti
- Nonostante la loro efficacia, è importante sottolineare che queste ci permettono di **mitigare** il problema solo in parte
  - Risolverlo del tutto è molto complesso e spesso impossibile
- A tal proposito, questa tematica è ancora molto trattata nella ricerca e resta ancora una **sfida aperta** per la comunità scientifica.

# Data augmentation

# Cosa possiamo fare quando abbiamo pochi dati?

- Oltre allo sbilanciamento dei dati, molto spesso sorge la necessità di dover ottenere molti più dati di addestramento
  - Addestrare un modello con molti dati lo rende più efficiente
- Tuttavia, non sempre abbiamo a disposizione grandi quantità di dati
  - Pensando ai dati medici, è raro averne molti a disposizione
- A tal proposito, sono sorte un gran numero di tecniche che ci permettono di generare questi dati in maniera sintetica
- L'insieme di queste tecniche viene chiamato “Data augmentation”



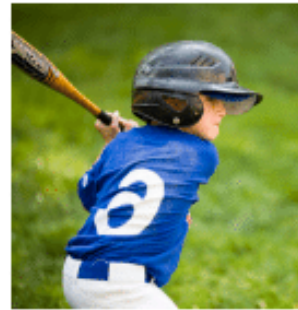
# Data Augmentation

- Le tecniche di Data Augmentation vengono utilizzate per aumentare l'ammontare di dati di addestramento
- Negli ultimi anni, l'applicazione di queste tecniche è divenuto un passo fondamentale durante lo sviluppo di un modello di ML
- La maggior parte di queste tecniche lavora partendo dal formato dei dati a disposizione. Tra le più conosciute troviamo:
  - **Simple Label-Preserving Transformations**
  - **Perturbation**
  - **Data Synthesis**

# Simple Label-Preserving Transformations

- Una delle tecniche più semplici ed utilizzate nella computer vision
  - Si **modifica** randomicamente un'immagine **preservando la sua etichetta**
  - Le modifiche possono prevedere operazioni di ritaglio, rotazione, inversione, cancellazione di parti dell'immagine, etc.
- Tutti i framework di ML più comuni prevedono il supporto di questa tecnica
  - PyTorch, TensorFlow, Keras,....
- Questa tecnica non viene solamente applicata alle immagini ma può essere **utilizzata anche per generare frasi** per task di NLP

# Simple Label-Preserving Transformations



A boy is holding a bat.

*computer vision*  
→  
*augmentation*



A boy is holding a bat.

A boy is holding a bat.  
Ein Junge hält einen Schläger.

*translation*  
→  
*augmentation*

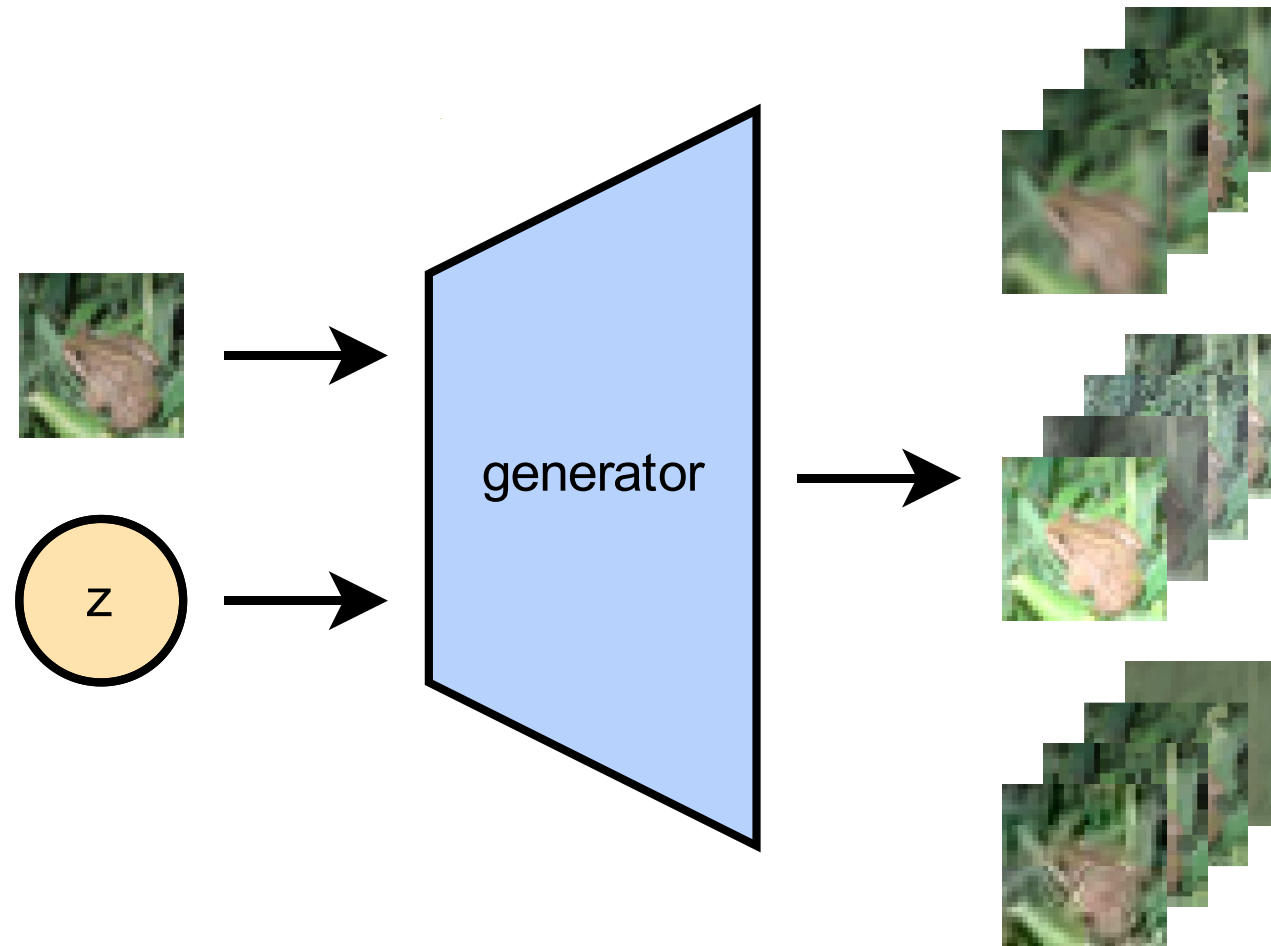
A boy is holding a **backpack**.  
Ein Junge hält einen *Rucksack*.

# Perturbation

---

- Altra tecnica di Data Augmentation label-preserving
  - Si aggiungono dati rumorosi a quelli già disponibili
  - Possiamo migliorare il decision boundary di un modello e migliorarne le performance.
  - Esistono diversi tipologie di tale tecnica: simple, adversarial,...
- Bisogna stare attenti quando si utilizza questa tecnica
  - L'aggiunta di rumore può anche rendere il modello meno performante
  - Si può sfruttare questa tecnica per attaccare i modelli
- Attacchi di questo tipo sono detti “adversarial attacks” e prevedono l'aggiunta di immagini rumorose nei dati per confondere i modelli

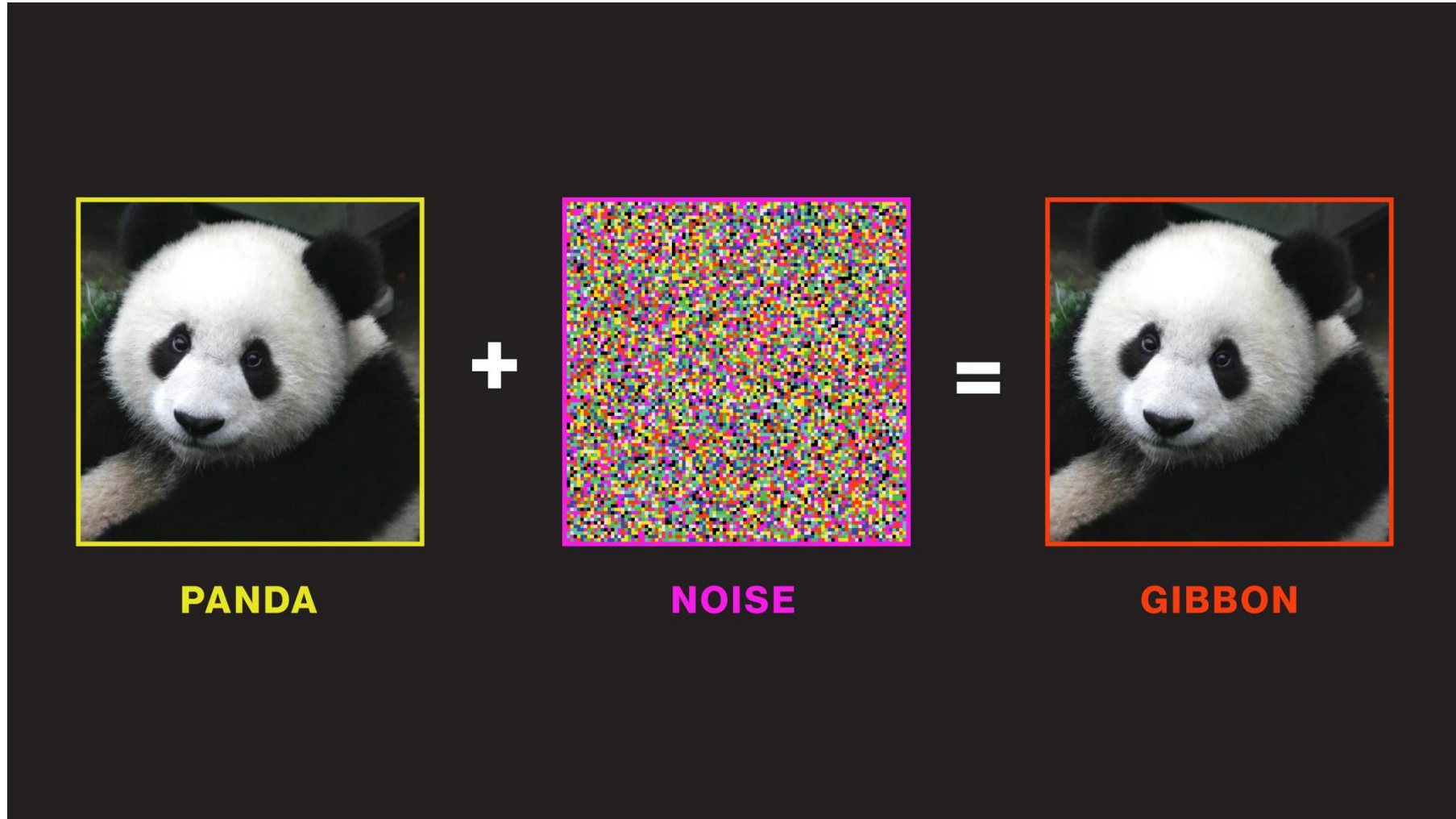
# Perturbation



# Adversarial augmentation

- Un tipo particolare di tecnica perturbation
  - Algoritmi che rilevano il minimo possibile noise injection necessario per causare una classificazione errata con alta confidenza
  - L'algoritmo DeepFool è un esempio di tecnica di questo tipo
- Algoritmi di questo tipo sono meno utilizzati con task di NLP
  - Anche aggiungendo rumore ad un image, il modello può comunque classificare bene l'immagine
  - Aggiungere caratteri ad una stringa, invece, può rendere quest'ultima incomprensibile
- Modelli di Linguaggio Naturale, come BERT, sfruttano questa tecnica come algoritmo d'apprendimento.

# Adversarial augmentation



# Data Synthesis

---

- L'ultima tecnica di data augmentation che vedremo riguarda i dati sintetici.
- Spesso ottenere dati reali è costoso e complesso
  - Ci vogliono tempo e risorse per collezionarli
  - Dobbiamo far fronte a problemi legati alla privacy
- E se usassimo **dati sintetici** invece che dati reali?
  - Sicuramente risulterebbe molto **più veloce e semplice** creare dei dataset
  - Possiamo generare la **quantità di dati che vogliamo in maniera automatica**
- Questa tecnica di data augmentation prende il nome di “**Data Synthesis**” e sta diventando molto popolare negli ultimi anni



# Data Synthesis

- Può essere utilizzata sia in task di NLP
  - Partendo da un template, possiamo generare frasi in maniera molto semplice

Template	Find me a [CUISINE] restaurant within [NUMBER] miles of [LOCATION].
Generated queries	Find me a <i>Vietnamese</i> restaurant within 2 miles of <i>my office</i> . Find me a <i>Thai</i> restaurant within 5 miles of <i>my home</i> . Find me a <i>Mexican</i> restaurant within 3 miles of <i>Google headquarters</i> .

- Sia in task di computer vision
  - **Mixup**: combinare campioni con label discrete per creare label continue
  - Supponiamo di avere  $x_1$  e  $x_2$  etichettati rispettivamente con 0 e 1. Possiamo generare  $x'$  combinando le etichette di  $x_1$  e  $x_2$  così:

$$x' = \gamma x_1 + (1 - \gamma)x_2$$


# Data Synthesis

---

- Tecniche come il Mixup sono in grado di migliorare le performance
  - Aumentano la capacità di generalizzazione del modello
  - Riducono i problem legati ad etichette corrotte
  - Aumentano la robustezza del modello contro gli “adversarial examples”
- Oltre questa tecnica, negli ultimi anni sono stati sviluppati sistemi basati sull’uso di reti neurali per la generazione di dati di questo tipo
  - Per esempio, CycleGAN permette di generare immagini per l’addestramento di un modello
  - Tuttavia, non c’è ancora un largo utilizzo di queste reti

# In conclusione

---

- In questa lezione abbiamo visto diversi aspetti legati ai dati di training
  - L'importanza di avere **dati di alta qualità**
  - La gestione di **problematiche** legate a questi dati
- Ancora oggi, i dati d'addestramento rappresentano le **fondamenta** per la progettazione di modelli di ML ad alte performance
  - È necessario quindi porre particolare attenzione ad essi
  - Dati di alta qualità  modelli di ML efficienti
- L'applicazione di tecniche di **campionamento, etichettatura, bilanciamento ed augmentation** sono, quindi, **essenziali** per pulire e migliorare questi dati