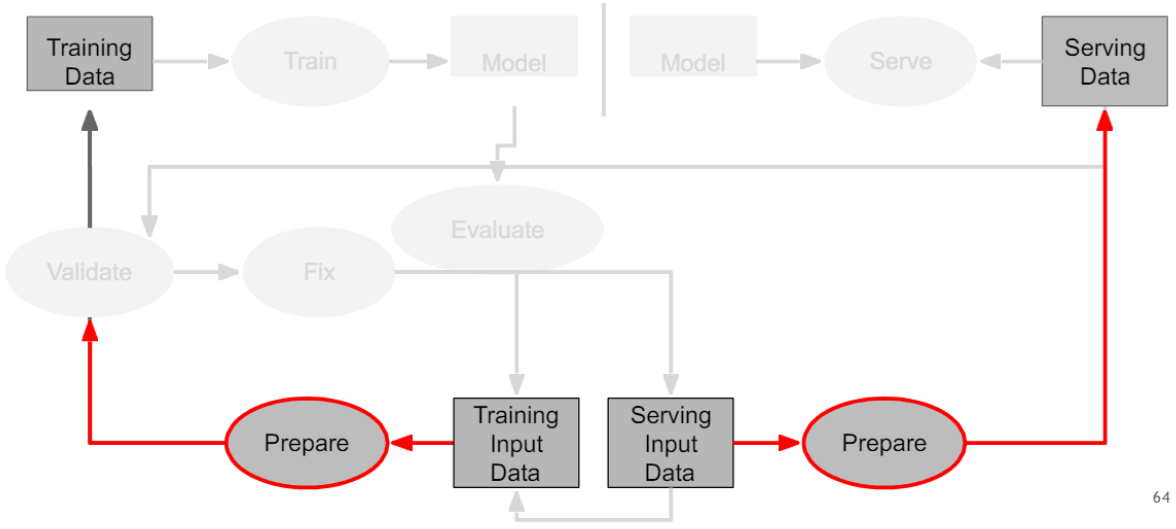
Lezione 10 22/04/2024

Data preparation

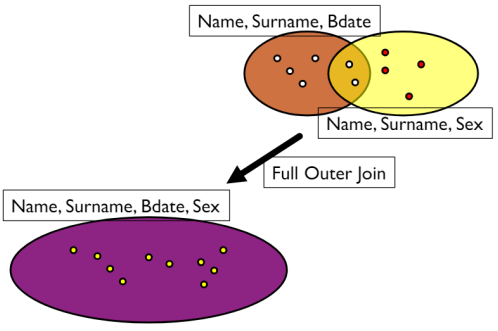
Dopo aver ricevuto i dati, questi vanno manipolati, trasformati. Le stesse operazioni vanno fatte sia sui dati di training che su quelli di serving.



Data integration

Da una parte ho un db con name, surname, bdate, e dall'altra ho un db con dati organizzati in modo diverso (+sex). Questi due vanno uniti.

In generale quindi ci saranno degli attributi in comune, ma anche attributi diversi. Può capitare che alcuni clienti siano anche appartenente ad entrambi.



Data integration è quando vogliamo integrare due dataset, e ci aspettiamo una tabella finale con tutti gli elementi della prima e della seconda tabella, con gli elementi presenti in entrambi rappresentati insieme. Lo schema finale

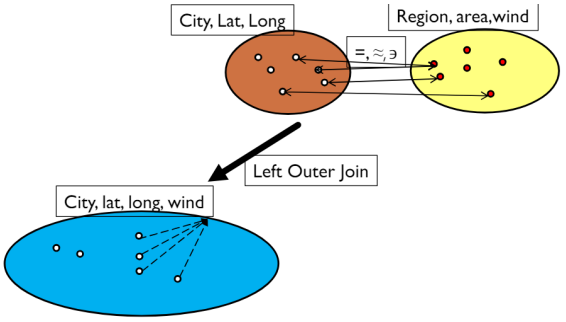
sarà rappresentato dall'unione insiemistica dei due.

Quindi ci saranno dei record con dei campi vuoti (bdate, sex).

Mi aspetto che le due tabelle descrivano le stesse cose, ma possibilmente possono avere delle feature leggermente diversi, come nell'esempio.

Data enrichment

Nel **data enrichment** ho una tabella principale a cui voglio aggiungere delle informazioni in più.



Può succedere che due città siano nella stessa regione, si può avere città senza regione.

Per esempio voglio fare delle previsioni metereologiche, ho le città con latitudine e longitudine, e voglio arricchire questi dati con le informazioni sul vento etc...

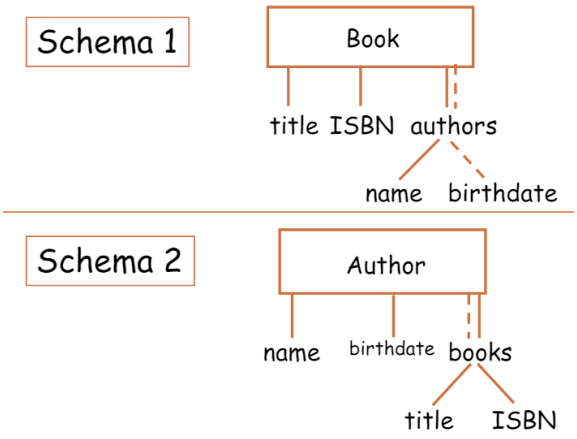
In questo caso ho una regione (quadrati), devo quindi usare una funzione di matching tra la regione e i dati di latitudine e longitudine.

La tabella finale conterrà tutti gli elementi della prima tabella, insieme a delle informazioni della seconda tabella quando ci sono.

Esempio libreria

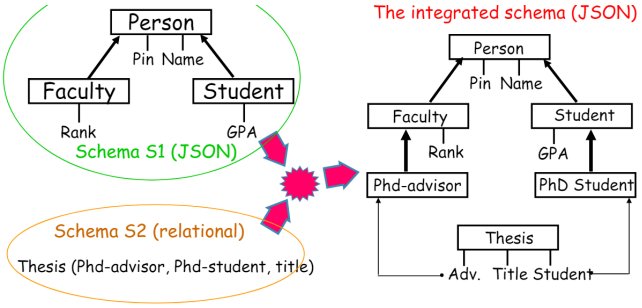
Voglio creare un dataset di una libreria, ma ho due schemi diversi. Uno rispetto ai libri e uno rispetto agli autori.

Bisogna quindi guardare bene gli schemi dei db e capirli bene.



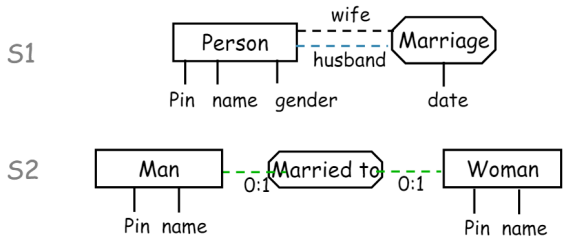
Esempio eterogeneo

La cosa si complica se ho uno schema relazionale e uno in json.



Esempio matrimoni

In questo caso il pin potrebbe essere diverso. Non esiste sempre la chiave primaria.

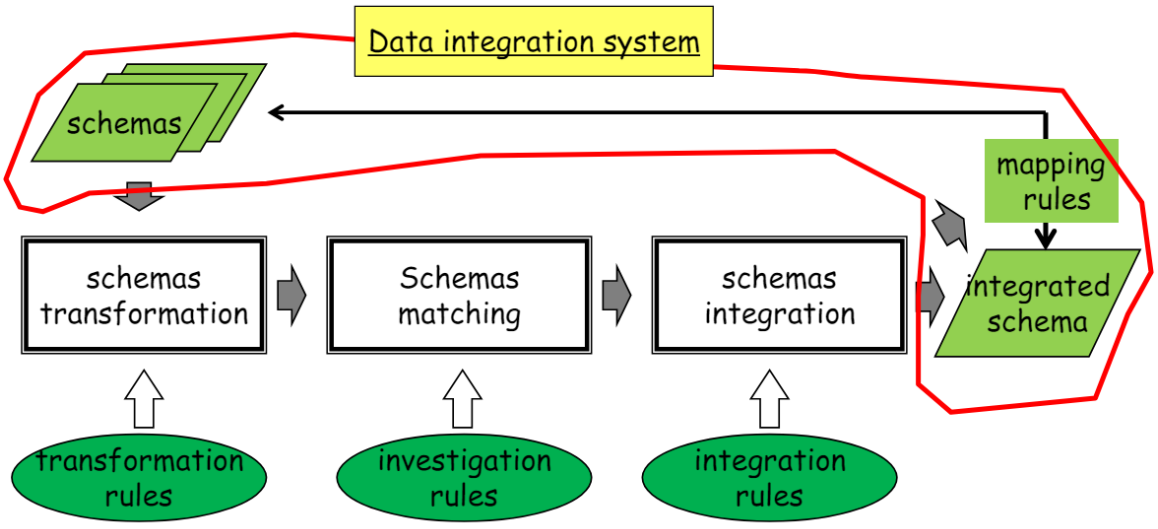


Quindi innanzitutto bisogna **capire lo schema**, e poi si può fare il merging. Ci possono essere **etrogeneità di modelli** (due modelli diversi, grafo, key value, etc) e **eterogeneità di nomi** (termini più specifici, meno specifici, nome diverso).

Potremmo avere domini diversi per gli stessi attributi (per esempio voti 0-30 o A-F).

Potremmo avere da una parte attributi propri e dall'altro attributi derivati (codice fiscale salvato o calcolato)

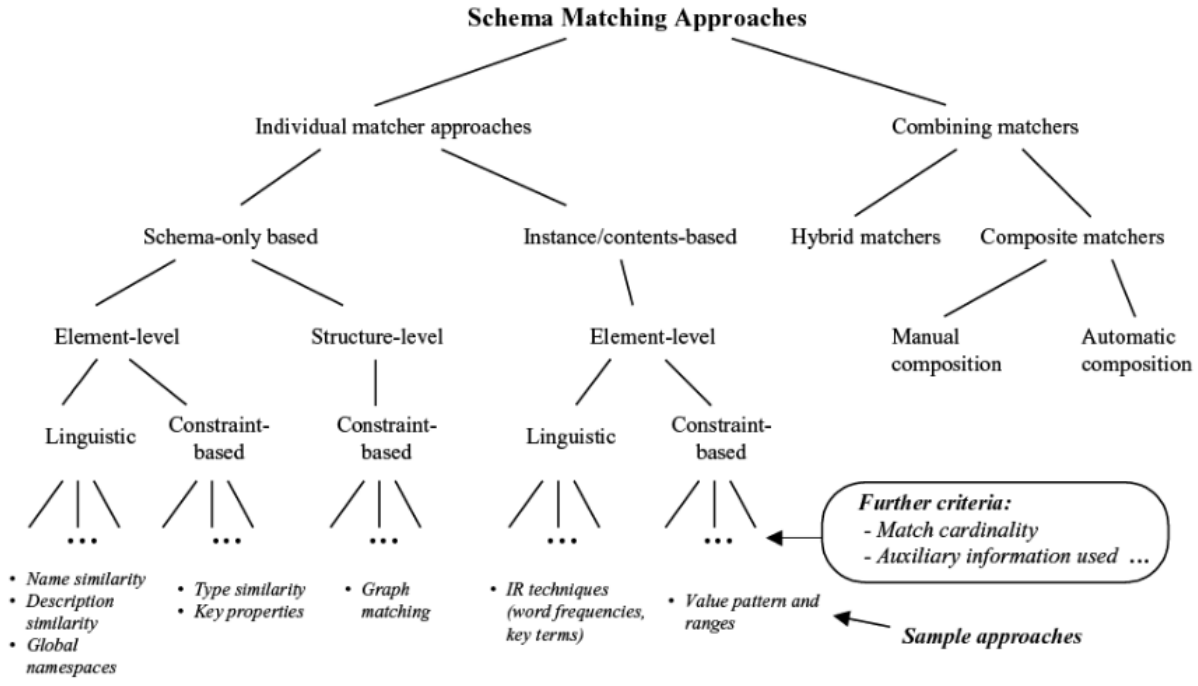
Linee guida



Nelle linee guida per fare data integration, prima bisogna capire quale sarà il formato finale.

Quindi bisogna fare schema **transformation**, **matching** e **integration**.

Ci possono essere approcci diversi:



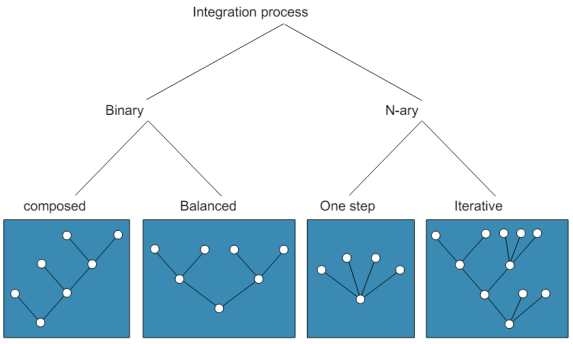
Fasi:

01. Schema transformation (or Pre-integration)
    - Input: n source schemas
    - Output: n source schemas homogeneized
    - Methods used: Model transformation + Reverse engineering
2. Correspondences investigation
    - Input: n source schemas
    - Output: n source schemas + correspondences
    - Method used: techniques to discover correspondences
3. Schemas integration and mapping generation
    - Input: n source schemas + correspondences
    - Output: integrated schema + mapping rules btw the integrated schema and input source schemas
    - Method used: New classification of conflicts + Conflict resolution transformations

---

Quando ho più dataset da integrare, posso avere l'approccio binario o quello n-ario.

Composto è quello che continua ad integrare i dati, partendo da una tabella e aggiungendo man mano dagli altri db.

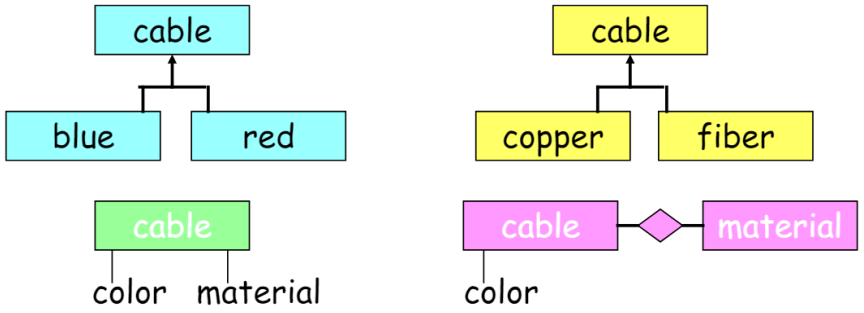


La soluzione **bilanciata** è quella dove per esempio ho un dataset di persone fisiche che hanno richiesto delle esenzioni perché sono ipovedenti, e lo vado ad integrare con il database delle persone che hanno patenti. Per esempio per trovare le persone che avevano la patenti e poi hanno perso la vista.

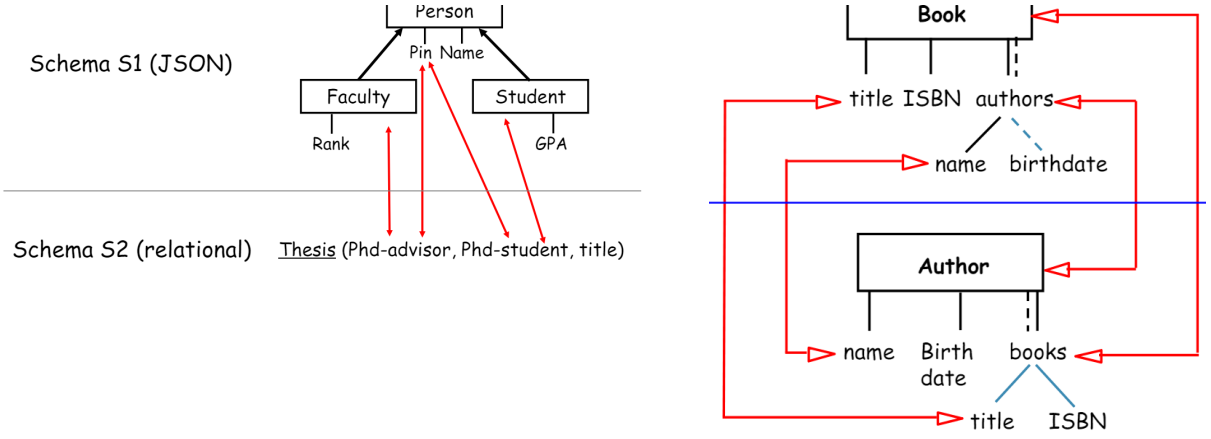
La soluzione **one step** fa tutto in un colpo solo, oppure ho **strategie miste** (per esempio se ho tanti schemi da mischiare).

Semantica

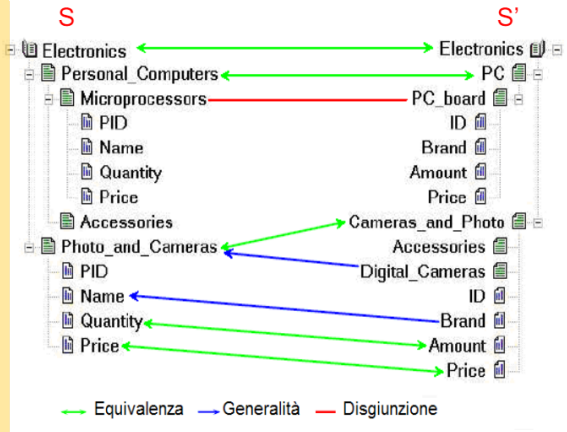
Per esempio posso definire un cavo in modo diverso, in base alla percezione del progettista.



Devo quindi essere in grado di trovare le corrispondenze.



Altro esempio:



Equivalenze sono uguali.

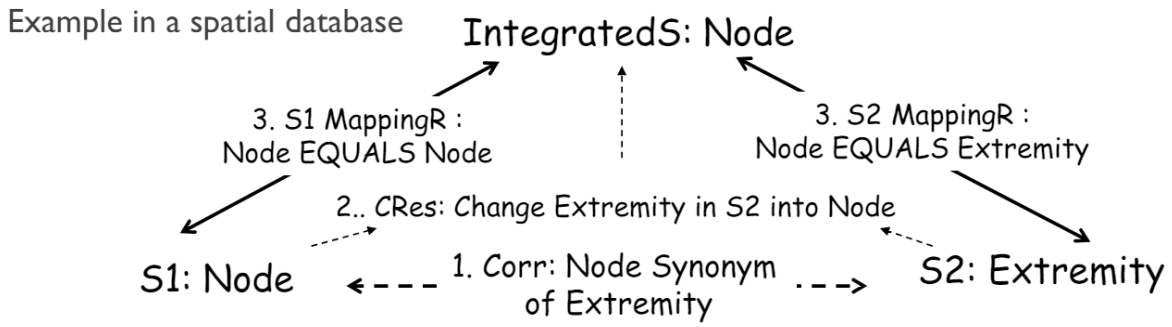
Generalità sono i termini più generici, che includono altri termini più specifici.

Disgiunzioni sono cose diverse.

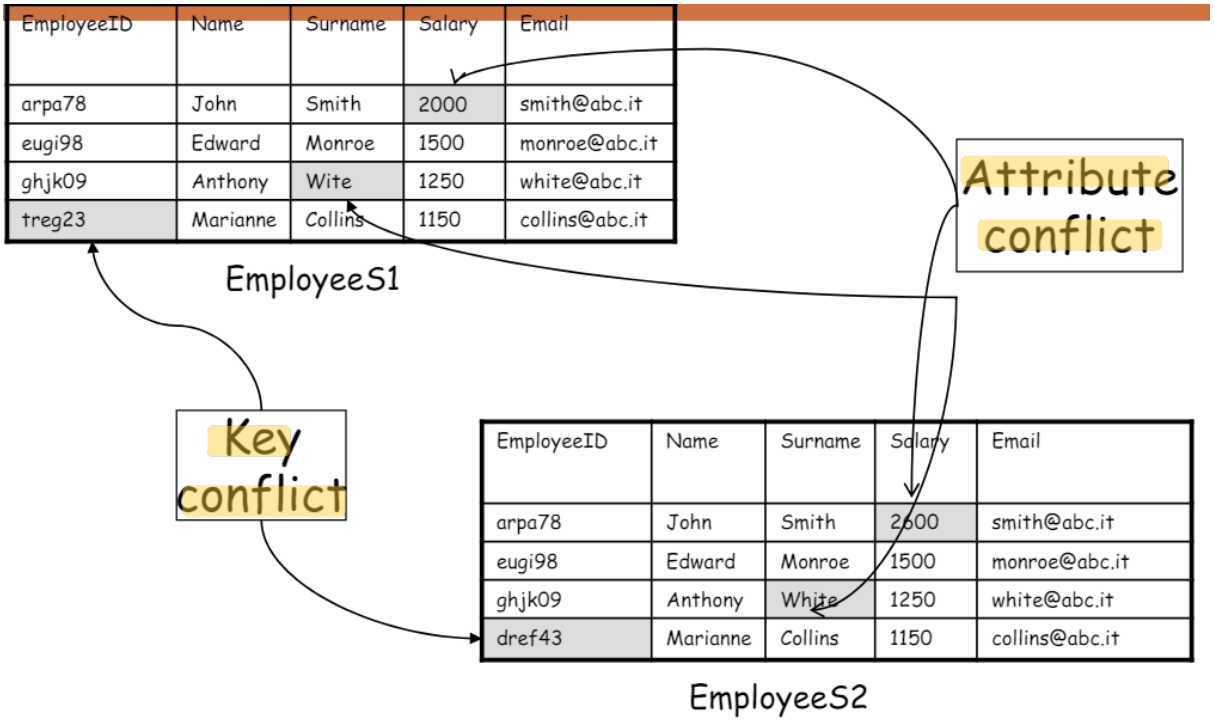
Bisogna quindi decidere come risolvere i conflitti, per esempio se terremo il nome amount o quantity.

Mapping rules

Alla fine avremo una struttura di questo tipo, definiremo gli attributi, le regole di trasformazione.



Dopo aver fatto il mapping degli schemi, posso aver dati duplicati o ancora peggio, dati diversi. Per esempio una stessa persona con dei dettagli diversi. Devo capire quale dei due è corretto, magari un db è più aggiornato dell'altro, magari uno contiene lo stipendio lordo e l'altro lo stipendio netto.



I key conflicts potrebbero per esempio essere due persone diverse. Bisogna capire quali sono gli elementi che caratterizzano la chiave primaria in entrambe le tabelle, magari sono generate random e quindi non si possono fare join sulle chiavi. Tutto dipende dalla semantica.

I conflitti sugli attributi sono da risolvere. Per esempio lo stipendio diverso, quale dei due è corretto? Vanno definite delle politiche, e vanno implementate

nel codice. La scelta dipende da cosa dobbiamo fare, in base all'obiettivo. Se per esempio decido di prendere sempre quello minore, devo rendere nota la **politica** di sottostimare gli stipendi.

A **design time** definiamo la **politica** che poi viene eseguita a **runtime**.

Le soluzioni base sono prendere tutti i dati da una tabella o dall'altra, oppure si possono applicare delle **funzioni** (minimo, massimo, medio, varianza, ...) l'importante è definirle bene nella documentazione, spiegando le motivazioni.

L'integrazione dei due dati può avere due forme: deduplicazione (integrazione dei dati all'interno di una tabella).

Deduplicazione

é un task dove vado ad individuare ed eliminare le descrizioni multiple di uno stesso oggetto nel db (es una persona salvata più volte).

Bisogna quindi capire se i record stanno davvero definendo lo stesso oggetto oppure no. Per esempio potrebbero esistere due persone con lo stesso nome e cognome.

Come regola potremmo usare per esempio che un record si riferisce allo stesso oggetto del mondo reale se differisce per una sola lettera in name & surname. Potrei decidere di ignorare le differenze linguistiche.

Name	Surname	Place of birth	Country
Crlo	Batini	Pescara	Italy
Karl	Batini	Pescara	null
Giulio	Batini	null	Italy
Carlo	Batini	Pscar	Italy

Integrazione

Rule of thumb: records are the same if **they differ only in one letter in Name & Surname**

Name	Surname	Place of birth	Country B
Crlo	Batini	Pescara	Italy
Giulio	Sandri	Milano	Italy
Leo	Siti	Milano	Italy
Gianni	Batini	Paris	null

Name	Surname	Place of birth	Date of B.
Carlo	Batini	Pescara	3-6-2007
null	Sandri	Roma	4-7-1998
Leonardo	Siti	Milano	24-7-2011
Gianni	null	Paris	13-3-1955

Tecniche esistenti

- Empirical e.g. "Crlo" in tuple 1.i is similar as "Carlo" in tuple 2.j; so, I match them since they are very close in terms of alphabetic symbols
- Probabilistic:
  - assume I know a sample of frequencies of "distances" between pairs of tuples
  - I may decide those matching and non matching ones in the universe by projecting such knowledge on the sample on the universe of pairs (see later)
- Knowledge based – decision based on rules that have to be matched by pairs of tuples
- Machine learning – use a ML task for detect if you rows represent the same real world object
- Mixed probabilistic and knowledge based

Devo utilizzare dei metodi per ridurre lo spazio di ricerca, si chiamano metodi di **blocking**. Vado quindi a fare dei confronti, ho bisogno di funzioni di comparazione che mi dicono quando due righe sono simili o no. La risposta può essere match, possible match, unmatch.

Steps dell'approccio probabilistico

- **Preprocessing**: bisogna formattare tutto nella stessa maniera, anche a livello di maiuscole/minuscole, di format delle date. I dati vanno **normalizzati**. Si può decidere di togliere le preposizioni, usare solo la lingua inglese, ...

Example

1. Via S. Giovanni 24
2. V. San Giovanni 24
3. V. S. Giovanni 24

Are all transformed into

Via San Giovanni 24

Example

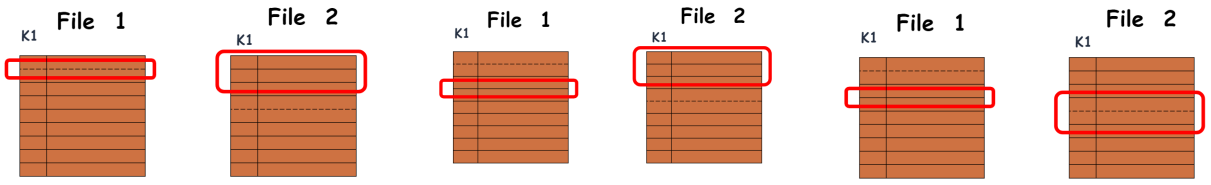
1. 21/12/2021
2. 12/21/2021
3. 13464675326

Are all transformed into

21/12/2021

- **Blocking**: ci sono varie tecniche, quella principale (blocking) prende un blocco di due tabelle, controlla tutti gli elementi del primo blocco con quelli della seconda tabella, e poi faccio scendere su entrambe le tabelle. Quindi ordino i blocchi per un certo attributo. Ci sono molti modi per ordinarli, per esempio il **sorted neighbour**. Come scegliere un buon attributo di ordinamento? Per esempio il genere non è un attributo forte. Il nome può andare bene, ma se commetto un errore nello scrivere prima lettera è un

problema (g/j, h, ...). Esistono delle metriche di similarità basate sul suono. Quindi servono delle **funzioni di comparazione**.



Esempio blocking con funzione:

Code	Given Name	Last Name	City of birth	Date of birth
034678	Carlo	Batini	Pescara	7/6/1949
543456	Giorgio	Dini	Crema	18/7/1987
543456	Maria	Domi	Milano	28/7/1987

Non matching

matching

Code	Given Name	Last Name	City of birth	Date of birth
BTNCRL	Crlo	Btini	Pesaro	7/6/1959
DNIGRG	Gioro	Din	Cremona	8/7/1987
DMIGRG	Giorgio	Dami	Modena	8/7/1994
DMISRG	Mria	Domini	Milano	18/9/1994

- **Compare**: bisogna definire una funzione che calcoli se due righe sono vicine o no.
- **Compare**: si prende un campione, si applica la funzione, basandosi sul campione si vede quali sono le soglie che trova un match di una coppia, a quale soglia non c'è match, e la zona grigia in mezzo.
- **Decide**: si valuta la distanza tra le coppie, e si scelgono i due tresholds dmin dmax.
- **Decide**: a questo punto si applica il sistema, l'approccio probabilistico. Dove la distanza d<dmin c'è match, dove d>dmax non c'è match, dove dmin<d<dmax c'è un possibile match.

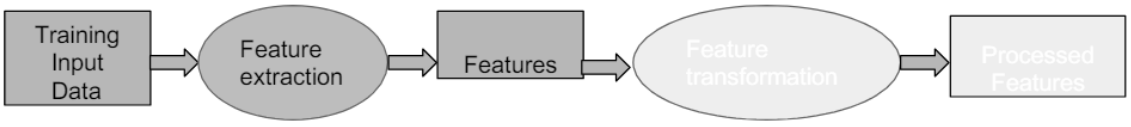
Feature engineering

è l'idea di trasformare il dato grezzo (la tabella) in un qualcosa che poi si da in mano al modello di machine learning. Quindi per esempio i dati testuali vanno trasformati in numerici per il modello.

A volte potrei dovere aggiungere nuove features, o anche nuove righe.

Con il **target transformation** si trasforma l'etichetta in un 0/1, un multiclasse, ... la trasformazione deve mantenere la semantica. In base a come trasformiamo la variabile target, questo impatta il risultato finale (immagine).

Per esempio vogliamo predire il prezzo medio di una casa, avendo la granularità del quartiere.



Objective: predict median housing price, at the granularity of city blocks.

Ho questi dati grezzi, ma se provo a fare un modello di classificazione, non funziona.

Bisogna fare grouping, splitting, trasformazioni, normalizzazioni, calcoli tipo medie, ...

- Standard set of techniques for feature transformation
  - Normalization
  - Bucketization
  - Winsorizing
  - One-hot encoding
  - Feature crosses
  - Use a pre-trained model or embedding to extract features

```
#Numerical Binning Example

Value      Bin
0-30   ->  Low
31-70  ->  Mid
71-100 ->  High

#Categorical Binning Example

Value      Bin
Spain  ->  Europe
Italy  ->  Europe
Chile  ->  South America
Brazil ->  South America
```

Vado a fare modifiche, controllando quando il modello migliora.

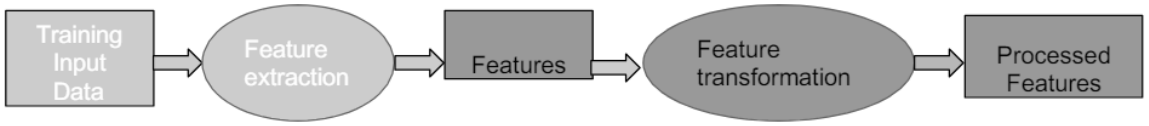
L'encoding sono le tecniche che trasformano le variabili categoriche in caratteristiche numeriche con informazioni più dettagliate. Se per esempio ho una tabella dove ad ogni utente associo una città, al posto di fare 0 roma, 1 madrid, 2 instanbul potrei fare così:

User	City
1	Roma
2	Madrid
1	Madrid
3	Istanbul
2	Istanbul
1	Istanbul
1	Roma

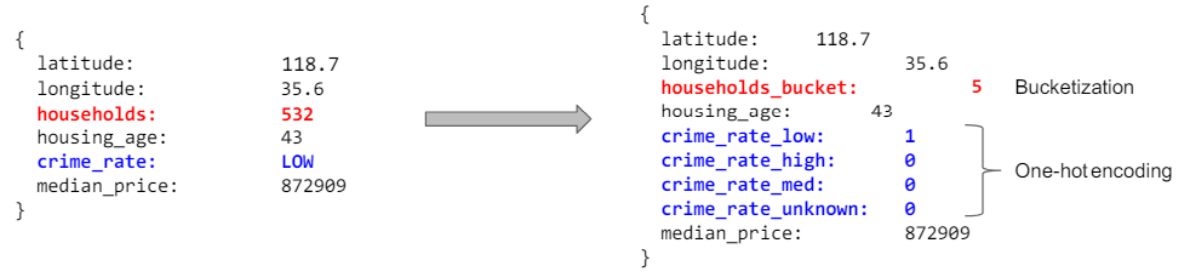
→

User	Istanbul	Madrid
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

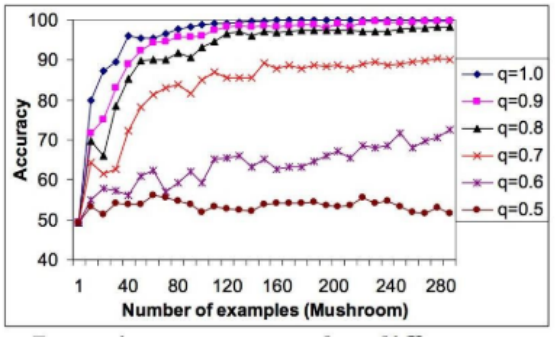
Qui oltre a dire che l'utente è stato in un posto, dico che che NON è stato negli altri, quindi aggiungo informazione.



Objective: predict median housing price, at the granularity of city blocks.



Aggiungere esempi costa, perché bisogna raccogliere più dati, serve un esperto che etichetta i dati. Ne vale il costo? Dipende dalla qualità dei dati, il bilancio dei dati, ...



Numeri di esempi del dataset e qualità, rispetto all'accuratezza del modello.

Quindi in realtà a volte bastano pochi campioni buoni per ottenere ottime performance.