Appunti Database

Brendon Mendicino

November 6, 2022

Contents

1 Introduzione

KDD: Knowledge Discovery from Data

Tecniche di data mining

- Regole di associazione: usate per trovare delle relazioni frequenti all'interno del database. Ad esempio: chi compra pannolini compra anche birra, il 2% degli scontrini contegono entrambe gli oggetti, il 30% degli scontrini che contengono pannalini contengono anche birra. Grazie alle regole di associazione si possono fare dei tipi di analisi come la basket analisys, ma puo essere utile anche per le raccomanadazioni.
- Classificazione: i classificatori predicono etichette discrete, esempio: nella posta elettronica alcune mail vengono segnate come spam. La classificazione definisce un modello per definire le predizioni, a volte non è sempre possibile creare dei modelli interpretabili ovvero dare una ragione per una determinata scelta.
- Clustering: gli algoritmi creano dei gruppi che raggruppano gli oggetti in esame, senza però dare delle motivazioni delle scelte effettuate.

2 Data Warehouse

Un DW è una base dati di supporto alla decisioni, che è mantenuta separatamente dalla base di dati operativa dell'azienda. I dati al suo interno sono:

- orientati ai soggetti di interesse;
- integrati e consistenti;
- dipendenti dal tempo;
- non volatili;
- utilizzati per il supporto alle decisioni aziendali;

Per la progettazione concettuale di un DW, non esiste un formalismo universale, il modello ER non è adatto ma viene invece utilizzata il modello **Dimensional Fact Model**.

Il DFM è composto da:

- Fatto: modella un insieme di eventi di interesse, che evolvono nel tempo (che può overe diversa granuralità).
- Dimensioni: sono gli attribuiti di un fatto, generalmento sono categorici.

- Misure: discrive una caratteristica numerica di un fatto.

Sulle dimensioni si possono definire delle gerarachie, che definiscono di fatto una dipendenza funzionale tra gli attributi, quindi di 1 a n. Ad esempio: **data** ha un arco **mese**, una data ha uno ed un solo mese (1 a n).

I costrutti avanazati sono:

- archi opzionali;
- dimensioni opzionali;
- attributo descrittivo: sono delle informazioni utili all'utente ma su cui non verteranno le interragazioni (ad esempio non si farà mai la group by su un indirizzo);
- non-additività: non si può fare la somma sulla metrica, il motivo è che non è modellato in modo tale da fare la somma;
- Fatto: fenomeno di studio;
- Misure: attributi del fatto;
- Dimensioni: tabelle collegate al fatto;

Schema a stella:

Snoflawke scheme: si esplicitano le dipendenze funzionali, questo però comporta un aumento delle operazioni di join.

Nella realtà lo snowflake è raramente utilizzato, il motivo è che il costo delle join può diventare oneroso. Un caso di utilizzo dello snowflake è quando si hanno dei dati condivisi.

Archi multipli:

Dimensioni degeneri: sono delle dimensioni con un solo attributo, questo si perchè nello stato attuale non si hanno delle specifiche per quell'attributo ma nel futuro si potrebbe facilmente estendere. Un'altra soluzione potrebbe essere un push down delle dimensioni degeri nella tabella dei fatti.

Junk Dimension: si può creare una dimensione che contenga tutte le dimensioni degeneri, le informazioni sono collegate semanticamente, è anche possibile unire delle informazioni scorrelate ma non è una scelta poco corretta, una soluzione potrebbe essere avere più junk dimensions.

3 Analisi

Sfruttando solo l'SQL è molto difficile fare delle analisi su un dw, infatti volendo calcolare delle operazioni per due argomenti diversi si devono fare più query. Estendendo il SQL si può, ad esempio, effettuare più operazioni leggendo una sola volta la tabella, ed effettuando il minor numero di join possibile.

Analisi OLAP I tipi di operazione sono:

- roll up: riducendo il livello di dettaglio del dato, ovvero eliminare una o più clausole della groupby o navigare la gerarchia verso l'esterno;
- drill down: si aumenta il livello di dettaglio oppure si aggiunge una dimensione di analisi;
- slice and dice: consentono di ridurre il volume dei dati selezionando un sottogruppo dei dati di partenza;
- tabelle pivot: come viene mostrato il dato;
- ordinamento: ordinamento in base agli attributi;

Queste operazioni possono essere fatte con più o una query.

3.1 Finestra di calcolo

Una finestra di calcolo fa dei calcoli a partire da una query sottostante, la finestra ha 3 operazioni sottostanti:

- partizionamento (**partition by**): partizionamnto dei dati, divide i record in gruppi a partire dall'attributo selezionato;
- ordinamento (**order by**): si definisce il criterio di ordinamento delle righe all'interno dei partizionamenti;
- finestra di aggregazione (**over**): porzione di dati, specifica per ogni riga di dato, su cui effettuare dei calcoli;

Example 3.1

Data la tabella Vendite(Città, Mese, Importo), calcolare per ogni città la media delle vendite per il mese corrente ed i due precedenti.

```
SELECT Citta, Mese, Importo,
    AVG(Importo) OVER (PARTITION BY Citta)
                        ORDER BY Mese
                        ROWS 2 PRECEDING)
    AS MediaMobile
FROM Vendite;
```

Quando la finestra è incompleta il calcolo è effettuato sulla parte presente, è possibile specificare che se la riga non è presente il risultato deve essere NULL.

Si può definire un intervallo fisico, superiore o inferiore.

```
    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
```

È possibile definire la tupla currente e quella che la precedono e che la seguono

```
ROWS UNBOUNDED PRECEDING (o FOLLOWING)
```

Il rggruppamento fisico è specifico per quando i dati non hanno delle interruzioni.

Per definire un intervallo logico si utilizza il costrutto **range**.

```
SELECT Citta, Mese, Importo,
    Importo / SUM(Importo) OVER () AS PerOverMax,
    Importo / SUM(Importo) OVER (PARTITION BY Citta) AS PerOverCity,
    Importo / SUM(Importo) OVER (PARTITION BY Mese) AS PerOverMonth
FROM Vendite
```

Se una **group by** è presente all'interno della query allora, tutte le entry che possono comparire nella finestra di calcolo sono solo quelle che compaiono nella group by.

Funzione di ranking La funzione di ranking serve a creare delle classifiche

- **rank()**: la funzione rank in presenza di più oggetti nella stessa posizione salta al prossimo record;
- **denserank()**: la funzione denserank tiene tutte righe con la stessa posizione;

...

```
SELECT Citta, Mese, SUM(Importo) AS TotMese,
    RANK() OVER (PARTITION BY Citta
                 ORDER BY SUM(Import) DESC)

FROM Vendite, ...
WHERE ...
GROUP BY Citta, Mese
```

Estensione della group by

- **rollup**: consente di calcolare le aggragazioni su tutti i possibili gruppi, eliminando una colonna alla volta, da destra verso sinistra, esempio: calcola le vendite per: (Citta, Mese, Prodotto), (Citta, Mese), (Citta):

  ```
  SELECT Citta, Mese, Prodotto, SUM(Importo) AS TotVendite
  FROM ...
  WHERE ...
  GROUP BY ROLLUP (Citta, Mese, Prodotto)

  ```

- **cube**: consente di calcolare tutte le possibili combinazioni del ragrruppamento;
- **grouping sets**: serve a definire degli aggregati su gruppi specifici definiti dall'utente;

3.2 Sintassi ORACLE

Raggruppamento fisico:

Example 3.2

Selezionare, separatamente per ogni città, per ogni data l'importo e la media dell'importo dei due giorni precedenti.

```
select citta, data, importo,
    avg(importo) over (partition by citta
                       order by data
                       rows 2 preceding
    ) as mediaMobile
from vendite
order by citta, data;

```

Raggruppamento logico:

Example 3.3

```
select citta, data, importo,
    avg(importo) over (PARTITION BY citta
                       ORDER BY data
                       RANGE BETWEEN INTERVAL '2'
                       DAY PRECEDING AND CURRENT ROW
    ) as mediaUltimi3Giorni
from vendite
order by citta, data;

```

Example 3.4

```
select COD_A, sum(Q) as sommaPerArticolo,
    rank() over (order by sum(Q) desc) as graduatoria
from FAP
group by COD_A

```

All'interno di oracle sono preseti delle funzionalità aggiuntive oltre alla funzione di rank, come:

ROW_NUMBER Assegno un numero progressivo ad ogni elemento in una partizione.

```
select tipo, peso,
    row_number over (partition by tipo
                     order by tipo)
```

```
from ...
where ...;
```

CUME_DIST Consente di calcolare le distribuzine cumulativa all'interno di una partizione, permette di definire un valore sulla distribuzione dei valori.

NTILE(n) Una funzione che da la possibilità di dividere le partizioni in sottogruppi

```
select tipo, perso,
    ntile(3) over (partition by tipo order by peso) as ntile3peso
from ...
where ...;
```

3.3 Esercizi

Cliente(CodCliente, Cliente, Provincia, Regione)
Categoria(CodCat, Categoria)
Agente(CodAgente, Agente, Agenzia)
Tempo(CodTempo, Mese, Trimestre, Semestre, Anno)
Fatturato(CodTempo, CodCliente, CodCatArticolo, CodAgente, TotFatturato, NumArticoli, TotSconto)

1. Visualizzare per ogni categoria di articoli

- la categoria
- la quantità totale fatturata per la categoria in esame
- il fatturato totale associato alla categoria in esame
- il rank della categoria in funzione della quantità totale fatturata
- il rank della categoria in funzione del fatturato totale

```
select categoria, sum(numArticoli),
    sum(totFatturato),
    rank() over (order by sum(numArticoli) desc),
    rank() over (order by sum(totFatturato) desc)
from fatturato f, categoria c
where f.codCatArticolo = c.codCat
group by categoria;
```

2. Visualizzare per ogni provincia

- la provincia

- la regione della provincia
- il fatturato totale associato alla provincia
- il rank della provincia in funzione del fatturato totale, separato per regione

```
select provincia, regione,
    sum(totFatturato) as fatturatoPerProvincia,
    rank() over (partion by regione
                 order by sum(totFatturato) desc
    ) as rankFatturatoPerRegione
from cliente c, fatturato f
where c.codCliente = f.codCliente
group by provincia, regione;
```

3. Visualizzare per ogni provincia e mese
   - la provincia
   - la regione della provincia
   - il mese
   - il fatturato totale associato alla provincia nel mese in esame
   - il rank della provincia in funzione del fatturato totale, separato per mese

```
select provincia, regione, mese,
    sum(totFatturato) as fatturatoPerProvinciaPerMese,
    rank() over (partition by mese
                 order by sum(totFatturato)
    ) as rankFatturatoPerMese
from cliente c, fatturato f, tempo t
where c.codCliente = f.codCliente and t.codTempo = f.codTempo
group by provincia, regione, mese;
```

4. Visualizzare per ogni regione e mese
   - la regione
   - il mese
   - il fatturato totale associato alla regione nel mese in esame
   - l'incasso cumulativo al trascorrere dei mesi, separato per ogni regione
   - l'incasso cumulativo al trascorrere dei mesi, separato per ogni anno e regione

```sql
select regione, mese,
    sum(TotFatturato) as fatturatoPerMese,
    sum(TotFatturato) over (
        partition by regione
        order by mese
        rows unbounde preceding
    ) as incassoCumulativoTot,
    sum(TotFatturato) over (
        partition by regione, anno
        order by mese
        rows unbounded preceding
    ) as incassoCumulativoPerAnno
from cliente c, fatturato f, tempo t
where c.CodCliente = f.CodCliente and t.CodTempo = f.CodTempo
group by regione, mese, anno;
```

4 Viste materializzate

Le viste materializzate sono necessarie per ridurre la lentezza della operazioni di group by per grandi moli di dati, le viste materializzate sono dei sommari precalcolati della tabella dei fatti.

Le VM usano con con costruto pricipale la group by, quando si crea una VM è convienente includere anche le dimensioni a granularità superiori, in modo da poter riutilizzare la tabella.

Per rappresetnare le dipendenze delle viste materializzate si utilizza un **reticolo multidimensionale**. Più ci si trova in alto al reticolo più ci si avvina alle dimensioni della tabella dei fatti, più si va in basso più si trova un granuralità maggiore.

La scelta delle viste viste tra tutte le possibili combinazioni è data da:

- si scelgie una sola vista da cui è possibile raggiungere tutti gli attributi;
- creo una vista per ogni query;
- scelgo delle viste intermedie che possono portare a ripsondere e più query;

4.1 Documentazione Oracle

Riducono i tempi di esecuzione delle group by e non si eseguono più le join. Nel DBMS Oracle esiste la **query rewriting**, che permette grazie all'ottimizzatore di interpretare le query e se i risultati corrispondono alle condizioni di creazioni delle viste, allora le query viene riscritta con la vista.

```
create materialized view NAME
[build {immediate|deferred}]
[refresh {complete|fast|force|never}
         {on commit|on demand}]
[enable query rewrite]
as
    QUERY
```

- immediate: lo schema della tabella viene popolata immediatamente, dato dallo schema di attributi presenti nella select;
- deferred: la vista viene creata, ma viene popolata successivamente;
- complete: i dati vengono presi interamente dal database;
- fast: i dati vengono presi in modo incrementale;
- force: se possibile viene eseguito il refresh in modalità fast, oppure in modalità complete;
- never: la vista non viene mai aggiornata;

- on commit: ogni volta che viene fatto un commit sulla tabella della query anche la vista viene aggiornata;
- on demand: viene definito dall'utente quando aggiornare la vista;
- enable query rewrite: abilita il dbms ad usare la vista per accellerary le query;

Per effetturare il refresh esistono dei tipi di job (a differenza del tipo di prodotto). Quando abbiamo bisogno del fast refresh, la tabella ha bisogno delle informazioni aggiuntive, ovvero dei file di log che ci informano delle nuove informazioni aggiunte al db, la **materialized view log** è associata ad una tabella che ha subito delle variazioni:

```
create materialized view log on
    TABELLA
with sequence, rowid
    (Attributo, ...)
including new values;
```

- squence: istante temporale in cui è avvenuta la modifica;
- rowid: indica la tupla che ha subito una modifiche;

Su queste keyword si deve difinire una lista di attributi da monitorare, si aggiunge **including new values** per supportare l'inserimento di nuove tuple.

5 Progettazione fisica

Fare una progettazione fisica comporta analizzare il carico di Si difiniscono delle strutture fisiche accessorie per velocizzare le operazioni. Si possono definire delle viste oppure degli indici, ad esempio: indici bitmap, indice di join, ...

La progettazione fisica è dipendente dal carico di lavoro.

La progettazione fisica è caratterizzata da una fase di tuning, utilizzata per testare gli indici e le viste create e decidere se matenerli o meno.

Gli inidici si possono creare sugli attributi che vengono selezionati più frequentemente, se il dominio è ridotto (come quelle categorici dei DW) si utilizza un a bitmap, altrimenti un B-tree.

6 Alimentazione dei Data Warehouse

Essendo dei dati derivati, la prima operazione da effettuare è l'ETL, se questo è complesso si va a definire un area di staging in cui il dato viene matenuto temporaneamente. Il processo di ETL va gestito sei per il popolamento del DW sia per quando verrà aggiornato con dati nuovi.

6.1 Estrazione

L'estrazione statica è la prima estrazione effettuata per popolare il DW. Per fare l'estrazione incrementale si possono:

- creare delle applicazioni ad hoc per i sistemi legacy;
- usando dei log, che non vanno ad interferire con il carico del db;
- usando dei trigger: sono proceddure che si attivano quando degli si effettuano delle operazioni specifiche;
- basata su timestamp: dove i recordi hanno il timestamp dell'ultima modifica effettuata su di essi;

6.2 Pulitura

Quando si effettua una estrazione ci si potrebbe trovare di fronte a:

- dati duplicati;
- dati mancanti;
- campo non previsto;
- valori errati o impossibili;
- inconsistenza del valore;

Ogni errore richede una tecnica specifica per essere risolto, le più comuni sono l'uso di dizionari per controllare errori di battitura, oppure il **join approssimato**, ad esempio: due database non hanno una chiave condivisa per identificare un utente dall'ordine effettutato, allora per fare una join si dovranno prendere i campi comuni, controllandone sempre la consistenza, oppure i problemi di **merge/purge**, ad esmpio: facendo il merge di due db le informazioni potrebbero essere duplicate ...

6.3 Trasformazione

Conversione dei dati nel formato di quelli presenti nel data warehouse.

6.4 Caricamento

In fase di caricamento i dati si caricano nel seguente ordine:

- dimensioni;
- fatti;
- indici e viste;

Problem 6.1 – Progettazione Magazzini

Tabelle:
Tempo(codT, data, mese, 3m, 4m, 6m, anno)
Magazzino(codMa, magazzino, citta, provincia, regione)
Modello(codMo, modello, categoria)
`UsoMtqMagazzino(codMa, codT, mtqLiberi, mtqTot)`
`UsoProdMagazzino(codMa, codMo, codT, numeroProdottiTotale, valoreTotaleProdotii)`
Query:

1. Relativamente al primo trimestre dell'anno 2013, considerando solo i magazzini della città di Torino,trovare per ogni coppia (magazzino,data) il valore complessivo di prodotti presenti in tale data nelmagazzino e il valore complessivo medio giornaliero di prodotti presenti nel magazzino nel corsodella settimana precedente la data in esame (data in esame inclusa):

```
select magazzino, data,
    sum(valoreTotProdotti) as valoreTot,
    avg(sum(valoreTotProdotti)) over (
        partition by magazzino
        order by data
        range between interval '7'
        day preceding and current row
    ) as valoreMedioSuGiornoCorrenteESettimanaPrecedente

from UsoProdMagazzino u, Tempo t, Magazzino m
where u.codT = t.codT and
    u.codMa = m.codMa and
    citta = 'torino' and
    anno = 2013 and
    3m = 1
group by magazzino, data;
```

2. Relativamente all'anno 2004, trovare per ogni coppia(città,data) la percentuale di superficie liberagiornaliera nella città. Associare ad ogni coppia un attributo di rank legato alla percentuale disuperficie libera giornaliera nella città (1 per la coppia con la più bassa percentuale di superficielibera giornaliera).

```sql
select citta, data,
    sum(mtqLiberi) / sum(mtqTot) * 100 as
   percentualeMtqLiberi,
    rank() over (
        order by sum(mtqLiberi) / sum(mtqTot) * 100
    ) as rankLowestPercentuale
from Tempo t, Magazzino m, UsoMtqMagazzino u
where t.codT = u.codT and
    m.codMa = u.codMa and
    anno = 2004
group by citta, data;
```

3. Relativamente ai primi sei mesi dell'anno 2014, trovare per ogni coppia (magazzino,data) la percentuale di superficie libera giornaliera.

```sql
select magazzino, data,
    100 * sum(mtqLiberi) / sum(mtqTot) as
   percentualeMtqLiberi
from Tempo t, Magazzino m, UsoMtqMagazzino u
where t.codT = u.codT and
    m.codMa = u.codMa and
    anno = 2014 and
    mese <= 6
group by magazzino, data;
```

4. Relativamente all'anno 2013, trovare per ogni coppia (magazzino,mese) il valore complessivo medio giornaliero di prodotti presenti.

```sql
select distinct magazzino, mese,
    avg(sum(valoreTotProdotti)) over (
        parition by magazzino, mese
    ) as valoreMedioGiornalieroComplessivo
from UsoProdMagazzino u, Tempo t, Magazzino m
where u.codMa = m.codMa and
    u.codT = t.codT and
    anno = 2013
group by magazzino, mese, data;
```

5. Relativamente all'anno 2015, trovare per ogni regione il valore complessivo medio giornaliero di prodotti presenti nella regione.

```sql
select regione, mese,
    avg(sum(valoreTotProdotti)) over (
        parition by regione, mese
    ) as valoreMedioGiornalieroComplessivo
from UsoProdMagazzino u, Tempo t, Magazzino m
where u.codMa = m.codMa and
    u.codT = t.codT and
    anno = 2015
group by regione, mese, data;
```

6. Relativamente all'anno 2014, trovare per ogni coppia(mese, regione) la percentuale di superficie libera giornaliera nella regione.

```
select regione, mese,
    avg (100 * sum(mtqLiberi) / sum(mtqTot)) over (
        partition by regione
    )
from UsoMtqMagazzino u, Tempo t, Magazino m
where u.codMa = m.codMa and
    u.codT = t.codT and
    anno = 2014
group by regione, mese, data;
```

7 Data Lake

I data lake sono dei repositori di dati, storicizzati per utilizzo futuro così come sono disponibili, in qualsiasi formato. Questi raw data potrebbero essere utilizzati in futuro.

I data lake danno la possibilità di storicizzare i dati per un uso futuro, inoltre tutti i dati sono contunuti in un repository comune, infatti è caratterizzato da bassi costi di storage e mantenimento, però può essere difficile estrapolare dei dati.

8 Data processing

Una collezione è costituita da oggetti di dato,

- **Attributo**: è una proprietà dell'oggetto;
- **Tipi di Attributo**: possono essere nominali, ordinali, intervalli, rapporti;
- **Proprità dei valori degli attributi**: possiamo definire equivalenza, ordine, addizione, moltiplicazione;
- **Attributi discreti e continui**: discreti hanno un nemero finito di valori, i continui hanno dei valori reali;

Tipi di dato da analizzare:

- record: sono i dati presenti in una tabella;
- grafi: come la struttura di una pagina web;
- ordinato: dati in cui esiste il concetto di squenza;

Esistono vari tipi di dato:

- **Document Data** Per ogni riga ho un doumento, per ogni colonna ho degli attributi che descrivono delle parole chivi all'interno del documento, ogni riga è un array che contiene la pesatura degli attributi, la pesatura può essre calcolata con algoritmi specifici;
- **Dato transazionale** Un dato transazionale è formato da un insieme di items all'interno della tabella, ogni transazione è identificata da un ID, nelle tabelle transazionali non esiste il concetto di ordine ne tra le riche, ne all'interno della transazione;
- **Dato a grafo**: ad esempio le pagine web hanno dei link ad altre pagine, potrei considere i link come gli archi del grafo, ognuno con un peso specifico, e la singola pagina come un nodo del grafo;
- **Qualità del dato**: nella maggior parte dei casi i dati presentano delgi errori, questi possono essere causati da rumori e outliers, dati mancanti o duplica, ...;
- **Outliers**: dati che escono al di fuori del comportamento medio e quindi molto rumorosi, l'abbiettivo dell'analisi di outlier detection è l'individuazione di questi dati e dell'eliminazione;

8.1 Data aggregation

Consente di combinare più record o più attributi, si fa questo per effettuare una riduzione della quantità di dati, ed avere dei dati più stabili con una vairiabilità minore.

8.2 Data reduction

Si possono effetuare due tipi di riduzione: riduzione degli attributi o riduzione dei valori. Per effettuare queste riduzioni esistono delle tecniche specifiche.

Una di esse è il **sampling** è una tecnica di statistica di analisi, ad esempio nella pipeline di datascience il sampling serve per trovare delle rappresentazioni adatte al dataset, facendo N esperimenti su un sample preso dal dataset si andranno a fare dei test sull'intero dataset per verificare che le ipotesi siano confermate, in figura si può vedere come il grafico col numero più piccolo di campia non sia più rappresentativo del dataset originale.

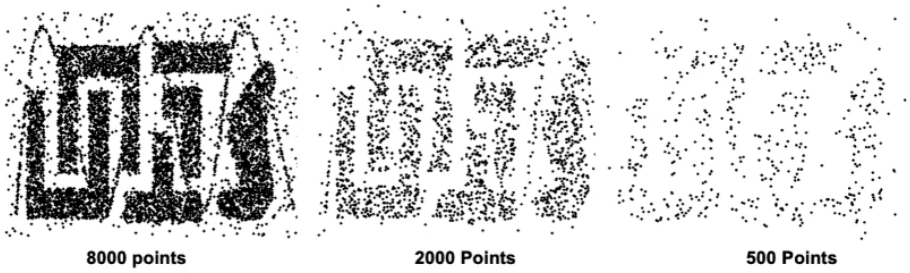

Figure 1: Sampling Example

I tipi di sampling possono essere:

- randomici: un dato estratto non viene reinserito;
- con rimpiazzo: un dato estratto può essere riestratto;
- sampling stratificato (più utilizzato): si può stratificare il dataset a partire da uno o più attributi (le partizioni vengono detti bucket), e poi vengono presi dei valori casuali;

Il problema con questi titi di approccio è che all'umentare delle dimensioni i dati diventano più distanti tra loro. Per ridurre gli effetti di avere troppe dimnesioni si possono applicare delle tecniche di riduzione dimensionale. Per questo motivo vengono utilizzate tecniche di **dimensionality reduction**, alcune di queste sono:

- **pca, svd, ...**: tecniche statistiche;
- **feature selection**: si vuole avere la proiezione del dato su un nuovo attributo in modo da aumentare la varianza, in qualche modo ridurre le feature ridondanti: le tecniche di feature selection sono: brute force; embedded approach (tramite un albero di selezione si selezionano solo i dati più significativi, può essere fatto quando gli attributi non sono elevati); filter (basiti sull'analisi di correlazione, per verificare se esistono delle correlazioni lineari); wrapper (viene utilizzato del data mining come black-box per identificare delle combinazioni di feature); feature creation (consiste nel combinare più variabili in una solo variabile, viene effettuato anche un processo di feature selection);
- **discretizzazione**: per effettuare una discretizzazione si deve mappare un valore continuo in un range di numeri discreti, una tecnica è quella di generare degli intervalli di uguale lunghezze e se la variabile ricade un in intrvalallo gli viene associato il simbolo corrispondente, questo potrebbe modellare dei volori outliers o rumorosi, si potrebbe anche usare del clustering, ovvero l'aggregazione di dati in base alla distanza tra vari valori, solitamente vengono usate due tecniche per validare i dati dopo la pipeline. Un caso particolare della discretizzazione è

la **binarizzazione** che consiste nel discretizzare una variabile e poi viene fatto il one-hot encoding (i valiri vengono mappati su una bitmap);

- **trasformazione**: un attributo va trasformato quando si vogliono riportare i valori in un'altra scala, una delle tecniche più comuni è la normalizzazine, ad esempio negli algoritmi di clustering viene definito uno spazio normato per calcolare la distanza tra i valori, le normalizzazioni più usate sono:

Theorem 8.1 – min-max

$$v' = \frac{v - min}{max - min}(new\_max - new\_min) + new\_min$$

Theorem 8.2 – z-score

$$v' = \frac{v - mean}{stand\_dev}$$

Theorem 8.3 – decimal scaling

$$v' = \frac{v}{10^j}$$

j intero più piccolo tale che: $\max(|v'|) < 1$

La similarità e la dissimilarità ci permattono di dire quando degli attributi sono simili o dissimili tra di loro, la similarità viene espresso in un intrevallo $[0, 1]$, con $1 =$ identici, per definire la similarità si definisce un concetto di distanza, ed una **matrice di similarità**, in cui ogni riga e colonna sono presenti i valori, ogni celle rappresenta le distanze tra i due valori. Le tre distanze utlizzate sono: Manhattan, Euclidea, Mikowski. In coso queste distanze non soddifisfino i criteri si definisc una distanza attraverso uno spazio vettoriale.

Una distanza importante è la **distanza di mahalanobis**, che mostra quanto due punti sono distanti in una distribuzione.

Theorem 8.4 – Mahalanobis

$$Maha(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

Σ matrice delle covarianze.

La correlazione è molto importante per trovare relazioni tra i dati. Una tecnica molto usata è trovare le combinazioni lineari attraverso il coefficente di pearson.

9 Regole di associazione

L'estrazione delle regole di associazione è un modo di trovare delle associazioni tra i volori prenti in un database transazionale. Per decidere queste associazioni solitamente si guardano le ricorrenze statistiche di valori comuni.

Una regola di associazione si definisce come:

Theorem 9.1 – Regola di associazione

$$A, B \implies C$$

Dove degli insiemi di oggetti (**itemset**) possono implicare altri insiemi.

- A, B = corpo della regola;
- C = testa della regola;

La freccia indica la **co-occorrenza**, indica che il copro è legato alla testa nelle transazioni del db. Per esempio: coca, pannolini $\implies$ latte.

Se si lavora con un db relazione possiamo estrarre una transazione associando ad ogni valore il suo attributo.

Definizioni:

Definition 9.1 – k-itemset

È un itemset che contiene k oggetti.

Definition 9.2 – support count

È la frequenza con cui una trasazione appare.

$$\#Roma, Colosseo = 2$$

$$sup(Roma, Colosseo) = 2$$

Data una regola di associazione $A \implies B$:

Definition 9.3 – Supporto

$$\frac{\#A, B}{|T|}$$

È la frazione di transazioni che contengono sia A e B. $|T|$ è la cardinalità del db.

Definition 9.4 – Confidenza

$$\frac{sup(A, B)}{sup(A)}$$

È la frequenza delle transazioni B che contengono anche A.

Per creare dei modelli per estrarre le relazioni, vengono definiti due paramtri che indicano la frequenza con quele frequnza devono apparire le relazione:

- supporto > **minsup** threshold;
- confidenza > **minconf** threshold;

Questo viene fatto per limitare il numero di relazioni che vengono estratte, perché nella maggior parte dei casi i db sono molto grandi.

L'estrazione si compone di due fasi:

1. si estraggono gli itemset frequenti, attraverso il vincolo sul supporto;
2. si estraggono le regole di associazione, attraverso il vincolo sulla confidenza;

Il passo più oneroso è l'estrazione degli itemset frequenti.

Il **candidato** è l'oggetto che potrebbe essere estratto, il **frequente** è l'oggetto che supera il valore di minsup. Per migliorare

9.1 Algoritmi Di Estrazione Di Itemsets

9.1.1 Apriori

Apriori si basa sul principio di quanto un itemset è frequnte, allora la porzione di lattice che ha come radice l'itemset allora si può esplorare il sottospazione dell'itemset, altrimenti no.

$$A \subset B \implies sup(A) \geqslant sup(B)$$

L'algoritmo di **Apriori** si basa sulla suddivide degli'itemset in livelli, in ogni livello si prendono dei candidati, facendo il join tra candidati frequenti (che superano la minconf) di livello k si generano candidati di livello k+1, per itemset che non rispettano il criterio di frequenza veiene fatto il pruning dell'albero delle scelte.

Sui candidati di lunghezza 2 va applicato il pruning (apriori). Alla fine si troverà l'insieme delle soluzioni che sarà l'unione di tutti gli itemset estratti.

Le limitazioni principali di questo algoritmo sono i costi di scansione del database, infatti dovrà essere letta più volte, inoltre se le transazioni sono molto lunghe l'algoritmo dovrà essere ripetuto n+1 volte (n = lunghezza transazoine), per superare questo limite si posso usare degli algoritmi per diminuire i costi legati alla lettura.

9.1.2 FP-Growth

Sono state proposte delle varianti dell'algoritmo per ottimizzare i problemi . Negli anni 2000 è stato proposto un nuovo algoritmo basato sulla memorizzazione. L'algoritmo **FP-growth**, instazia in memoria un albero dove si trova la proiezione del database originale che considera gli item che soddisfano la soglia di supporto, una volta creata la struttura di supporto in memoria non si accede più in memoria secondaria, ottimizzando la lettura degli item. La struttura in memoria prende il nome di **FP-tree**. L'algoritmo funziona nel seguente modo:

1. vengono contati gli item e vengono scartati quelli con sotto la soglia;
2. viene creata una header table con gli item in orine descrescente rispetto al supporto;
3. viene scansionata per l'ultima volta il database, le transazioni vengono ordinate in base alla header table;
4. ogni scansione di una transazine ordinata, viene preso l'item ed inserito nell'albero, ogni nodo contiene l'item ed il numero di volte che è stato trovato per ogni transazione letta;
5. l'inserimento nell'albero viene fatto a partire dall'ordine degli item nella transazione (simile algi alberi formati a partire da ogni lettera di di parole), ogni volta che si passa da un nodo già inserito il suo contatore aumenta;
6. è importante collegare la header table ai nodi nell'albero, questo viene fatto attraverso una **node link chain**: l'header punta ad un nodo con lo stesso item, quando percorrendo l'abero si trova un altro item il nodo precedente avrà come nodo successivo il nodo corrente;

Viene poposto anche un algoritmo di visita per esrarre gli itemset:

1. viene letta la header table dall'item col supporto più basso;
2. viene creato un **conditional pattern base** di un item, una proiezione dell'fp-tree condizionato ad un item;
3. il CPB viene visitato ricorsivamente per travore i candidati;
4. ad ogni nodo visitato viene recuperato il path dell'albero fino ad esso, se il supporto del nodo è minore del minsup il path viene scartato, e si passo al prossimo nodo della node-link chain;

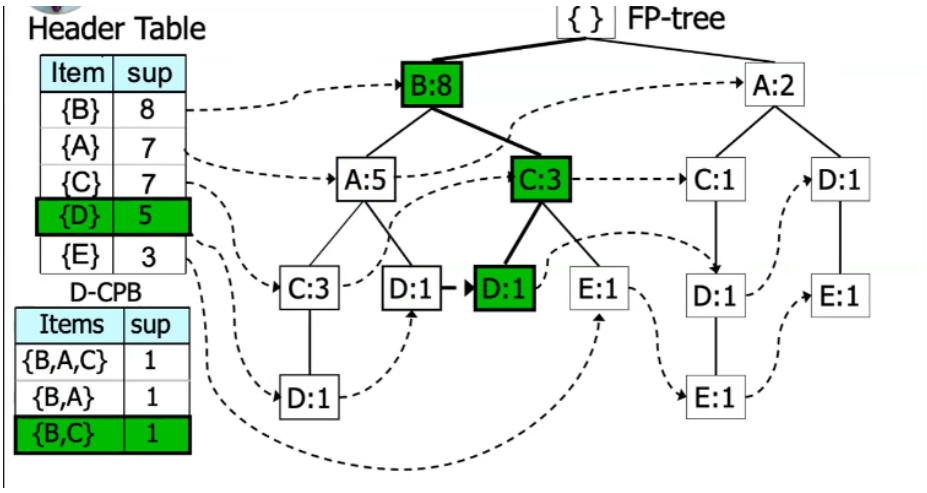

Figure 2: Cpd di D al terzo step

5. data la cpb va creata una Conditional header table, da questa header table, si crea un'altro fp-tree, questo si applica ricorsivamente, ogni chiamata è condizionata (ovvero la cpd sarà preceduta dall'itemset del chiamante: D -¿ DC -¿ ...);

6. quando non si riesce a creare una header table si torna al chiamante e si passa alla entry superiore nella header table del chiamante;

7. prima di creare la nuova cpb, si prende l'itemset che arriva dal chiamante e se il valore nell'header tabel del item corrente è maggiore del minsup allora l'itemset concatenato all'item corrente viene insireti negli itemset frequenti;

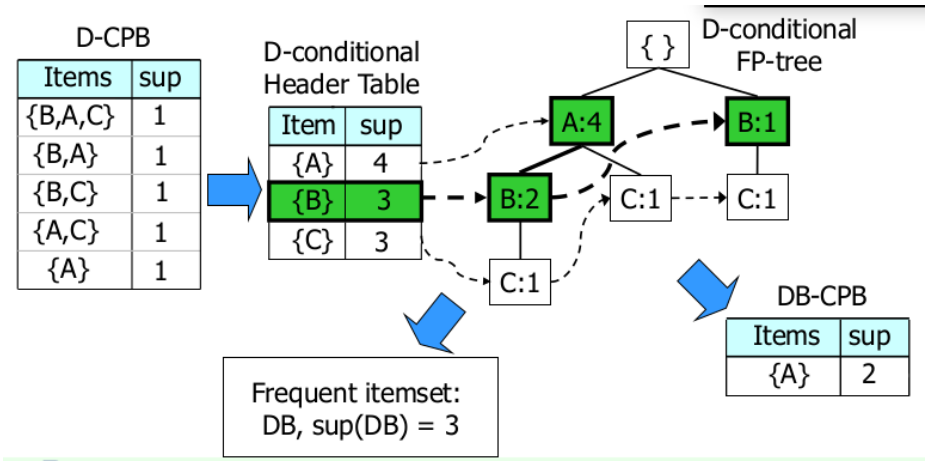

Figure 3: Esempio Aggiunta Itemset Nei Valori Frequenti

Questo algoritmo funziona molto bene se la memoria non viene saturata (bottleneck).

Un altro problema sono l'alto numbero di itemset che vengono estratti, infatti anche con db molto piccoli si possono trovare un numero molto elevato di iteset frequenti, per questo si opta per i **itemset massimali frequenti**, IMF per definizioni sono gli itemset che hanno il massimo numbero di figli, e sono unici nel loro sottolivello, oppure esistono i **closed itemset** che sono gli itemset con nessuno dei suoi immediati superset hanno lo stesso valore.

9.2 Effetto delle soglie

La scelta dei valori di supporto deve essere idoneo, infatti con un valore troppo basso non si identificano delle relazioni e con valori troppo alti emergono relazioni molto deboli. Anche la confidenza comporta dei problemi, se il valore di cui si calcola la confidenza ha un valore molto ampio, si rischiano di ottenere valori errati, per evitare di incrociare queste informazioni si utilizza la regola del lift.

Definition 9.5 – Lift

Dato $r : A \implies B$, allora la Correlazione o lift è:

$$C = \frac{P(A,B)}{P(A)P(B)} = \frac{conf(r)}{sup(B)}$$

Un esempio di regola di associazione potrebbe anche essere l'aggregazione di dati, andano ad accoppirare una tassonomia ai valori, possiamo, aggregano gli attribuiti, vedere il loro supporto crescere, rappresetando in modo generalizzato un comportamento, per andare a soddifare un servizio.

10 Classificazione

Le classificazioni cercano, attraverso dei modelli di assegnare dei tag ai dati, attraverso delle tecniche supervised (vuol dire che abbiamo già a disposizione un pool di dati da cui possiamo estrapolare le informazioni per assegnare un tipo di tag).

Per applicare la classificazione si ha bisogno di dei dati di training che hanno gia dei tag con il quale si va a generare un modello, per classificare dei nuovi dati si parte dandoli in pasto al modello e partendo dai volori degli attributi si generano delle nuove etichette.

Per poter realizzare un modello di classificazione si ha bisogno di dati di training, usati per generare il modello, e dati di test, usati per validare il modello, ognuno di questi dati ha già associato ad essi una classe di tag. Una volta che si trova un modello adatto, si può insireri in un applicazione per predirre i tag.

Gli algoritmi che generano i modelli hanno delle caratteristiche:

- accuratezza;
- interpretabilità;
- incrementalità: il modello può essere aggiornato all'arrivo di nuovi dati;
- efficenza;
- scalabilità: performance dell'algoritmo rispetto al numero di dati;
- robustezza: capcità dell'algoritmo di operare in presenza di dati rumorosi o mancanti;

10.1 Alberi di Decisione

Attraverso un albero di decisione, dati i dati di input è possibile inferire la classe di etichetta.

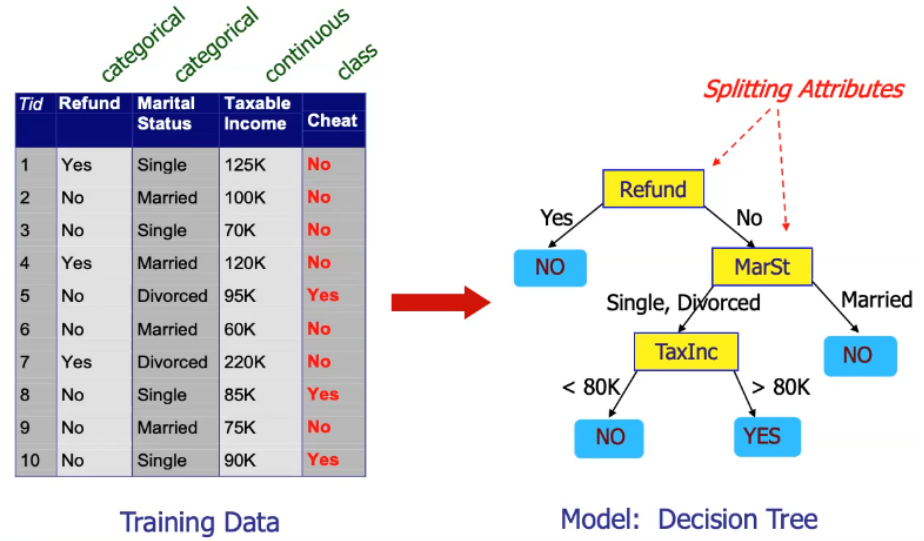

Figure 4: Albero Di Descisione

In un albero di decisione le folgie corrispondo all'etichetta di classe che satà assegnata alla entry. Questo modello dipendo molto dai dati, infatti se gli attributi cambiano anche il modello va modificato opportunamente. Per generare gli alberi di decisione esistono degli algoritmi, un di questi è l'**algoritmo di Hunt**. L'algoritmo parte dal leggere il database, e si cerca di individuare l'attributo che ha la maggiore capacità di dividere il db in due sottogruppi, il primo passo è duqnue individuare l'attributo che riesce a dividerere il db in due gruppi più omogenei possibili rispetto al tag. Successivamente si cerca il prossimo attributo che effettua il prossimo miglior partizionamento. Questo albero non è aggiornabile all'arrivo di nuovi dati.

Per effettuare lo split bisogna partire dai tipi di dati con cui si lavora:

- attributo categorico: si possono effettuare n-split diversi per ogni valore, se il tag è binario allora i valori vengono partizionati in due gruppi;
- attributo numerici: si può usare la discretizzazione, oppure si può usare una condizione di test;

Per stimare la purezza (grado di omogeneità) dei nodi che vengono generati, esistono delle metriche per calcolare l'impurità: gini index, entropia, missclassification index. Il primo caso è quello di utilizzo del **Gini index**, con questo metodo viene misurata l'impuità prima e dopo lo split, il gini index si misura con:

$$GINI(t) = 1 - \sum_j (p(j|t))^2$$

$p(j|t)$ = frequenza della classe j al nodo t. Più il gini index si avvicina allo zero, più la classe è impura, più il valori si avvicina a $(1 - \frac{1}{\text{numro di classi}})$ più il nodo è impuro, viene

preso l'attributo che genera il gini index col valori più basso. Nelle implementazioni reali vengono fatti dei test utilizzando metriche diverse con successive validazioni.

Per la creazione di un albero va definito anche un criterio di stop:

- ...
- pre-pruning: se un nodo è quasi puro rimuovo non scendno più nell'albero;
- post-pruning: generato l'albero completo vado a tagliare i rami dell'albero con delle caratterisiche troppo specifiche;

11 Trigger

Consideriamo un esmpio di tema di esame.

a:

```
misure: sum(incasso), sum(#consulenze)
tabelle: INCASSO, TEMPO, SERVIZION, SEDE-CONSULENTI
gb: semestre, tipologia-servizio
selezione: regione = 'Lombardia'
```

b:

```
misure: sum(incasso), sum(#consulenza)
tabella: INCASSO, SEDE-CONSULENTI, SERVIZIO, TEMPO, AZIENDA
gb: regione, servizio, anno
selezione: Nazionalita = 'Italia' or
    Nazionalita = 'Germania'
```

c:

```
misure: sum(incasso), sum(#consulenze)
tabelle: INCASSO, SEDE-CONSULENTI, SERVIZIO, TEMPO
gb: tipologia-servizio, regione, semestre
selezione: anno >= 2017 and anno <= 2019
```

Date le query precedenti si crei una vista materializzata:

```
-- Query blocco A
select servizio,
    tipologia-servizio,
    semestre,
    anno,
    regione,
    nazionalita,
    sum(incasso),
    sum(#consulenze)
from incasso i, tempo t, azienda a, servizio s, sede-consulenze sc
where condizioni di join
group by servizio,
    tipologia-servizio,
    semestre,
    anno,
    regione,
    nazinalita
```

L'identificare minimale sarà: (servizio, semestre, regione, nazionalita)
Punto 2: ...
Punto 3:

```
insert into viewIncassi(servizio,
    tipologia-servizio,
    nazionalista,
    semestre,
    anno,
```

```
    regione,
    incassotot,
    numconsulenzetot)
(blocco A)
```

Punto 4:

```
create trigger refreshViewIncassi
after insert on incasso
for each row
declare
    varServizio varchar(20);
    varTipologiaServizio varchar(20);
    varSemestre varchar(20);
    varAnno varchar(20);
    varNazionalita varchar(20);
    varRegione varchar(20);
    n int;
begin
-- leggere le tabelle dimensionale x recuperare
-- i valori dell'identificatore della vista materializzata:
-- servizio, nazionalita, semestre, regione
select servizio, tipologia-servizio into varServizio,
    varTipologiaServizio
from servizio
where idServizio = :NEW.idServizio;

select semestre, anno
from tempo
where idTempo = :NEW.idTempo;

select nazionalita into varNazionalita
from azienda
where idCategoriaAzienda = :NEW.idCategoriaAzienda

select regione into varRegione
from sede-consulenti
where idSede = :NEW.idSede;

-- verifico se esiste una tupla in viewIncassi con
-- associati i valori estratti

select count(*) into n
from viewIncassi
where nazionalita = varNazionalita and
    semestre = varServizio and
    servizio = varServizio and
    regione = varRegione;

if (n > 0) then
    update viewIncassi
    set incassoTot = incassoTot + :NEW.incasso
```

```
        numConsulenze += :NEW.#consulenze
    where servio = varServizio and
        nazionalita = varNazionalita and
        semestre = varSemestre and
        regiono = varRegione;
else
    insert into viewIncassi ( ... , )
    values (varServizio,
        varTipoServizio,
        varNazionalita,
        varSemestre,
        varAnno,
        varRegione,
        :NEW.incasso,
        :NEW.#consulenze);
end if;
end;
```

Punto 5:

```
create trigger updateViewIncassi
after update of tipologiaServizion on servizio
for each row
declare
    typeServizio varchar(20);
begin


end;
```

Punto 6:

```
create materialized view log on incasso
with sequence, row id
(...)
including new values;

create materialized view log on servizio
with sequence, row id
( ... )
including new values;

create materialized view log on tempo
with sequence, row id
(. ...)
including new values;
```