Elaborato K-Anonimity

Lorenzo Molinari

Aprile 2024

Contents

1 Introduzione

Nel mondo delle reti globali, le informazioni e i dati condivisi, in forma grezza, tabellare, statistica e in molte altre, assumono un'importanza crescente. Ogni giorno, vengono raccolti online dati personali che spesso possono essere ricondotti a dei rispondenti (ovvero a individui specifici). Per questo motivo, è fondamentale che tali dati vengano protetti e anonimizzati, poiché attributi come data di nascita, genere, numeri di previdenza sociale, stato famigliare, salari, codice postale e molti altri potrebbero essere collegati a persone fisiche identificandole ovvero de-anonimizzando la loro identità.

Per proteggere la privacy degli utenti, sono stati sviluppati diversi approcci, come la crittografia o la rimozione di questi attributi. Un metodo efficace per garantire l'anonimato mantenendo l'integrità dei dati è il K-anonimato. Questo approccio, basato su una metrica che misura il grado di privatezza di una base di dati, cerca di anonimizzare un database in modo che ciascun individuo, i quali dati sono presenti nella tabella, non possa essere distinto da almeno k altri individui, riducendo o eliminando così il rischio di re-identificazione.

1.1 Macrodata e MicroData

Nell'ambito business, i dati possono essere divisi in due sotto categorie: macrodata e microdata

1.1.1 Macrodati

I macrodati rappresentano un insieme di dati pubblicati a fini statistici, pubblicati in forma aggregata per preservare la privacy degli individui i cui dati sono stati raccolti. Rispetto ai microdati, che forniscono informazioni più specifiche riguardo gli utenti, i macrodati danno minori preoccupazioni dal punto di vista della privacy.

I macrodati possono essere suddivisi in due tipologie principali:

1. *Tabelle di conteggio (o frequenza)*: In cui ogni cella di queste tabelle contiene il numero di soggetti (quindi il conteggio) o la percentuale di rispondenti (la frequenza) che soddisfano una specifica caratteristica richiesta.

2. *Tabelle di grandezza*: In cui ogni cella di queste tabelle contiene un aggregato di interesse.

La pubblicazione dei macrodati può avvenire secondo due modalità:

1. *Database a fini statistici*: In questi database i DBMS rispondono solo a query statistiche (fornendo dati aggregati). Tuttavia, poiché le query sono scelte dall'utente, è necessario un controllo per evitare la possibilità di ottenere dati individuali. Inoltre è importante sottolineare che più utenti possono combinare le loro conoscenze (pratica chiamata collusione), portando a potenziali rischi per la privacy degli individui.

2. *Dati a fini statistici*: Questi dati consistono in statistiche precomputate sulle informazioni da pubblicare. Le query vengono eseguite off-line e non dai singoli utenti, riducendo così il rischio di esposizione dei dati. Il controllo per evitare problemi legati alla conoscenza dei dati da parte degli utilizzatori viene eseguito prima del rilascio degli stessi.

1.1.2 Microdati

I microdati sono un insieme di dati pubblicati che contengono informazioni specifiche per un certo gruppo di utenti presenti nel database. Negli anni, la loro diffusione è in costante aumento poiché il rilascio di microdati rispetto a dati statistici pre-computati consente maggiore flessibilità nell'utilizzo dei dati e una ampia disponibilità delle informazioni. Questo incremento dell'accessibilità dei dati, tuttavia, potrebbe esporre i microdati a rischi legati alla privacy degli utenti, diventa perciò fondamentale l'adozione di tecniche in grado di garantire l'anonimato e la protezione delle informazioni.

1.2 Attributi

Gli attributi presenti in un database possono essere categorizzati in questo modo:

- *Identificatori*: attributi che identificano in maniera univoca un determinato soggetto.
- *Quasi-Identifier*: insieme di attributi che possono essere associati ad altri attributi provenienti da sorgenti esterne che permettono l'identificazione dei soggetti a cui si riferiscono o riducono l'incertezza riguardo alla loro identità. Più questi sono dettagliati, più espongono gli utenti presenti nella base di dati. La scelta di quali attributi includere in questo insieme è spesso molto difficile e comprende molti aspetti. L'approccio più conservativo, ad esempio, include come quasi-identificatori tutti gli attributi che potrebbero essere o diventare campi che permettono l'identificazione dell'identità degli utenti.
- *Confidenziali*: attributi che contengono informazioni sensibili dell'utente a cui si riferiscono.
- *Non confidenziali*: attributi che non contengono informazioni sensibili dell'utente a cui si riferiscono; il loro rilascio non conduce a nessuna problematica.

Una cattiva gestione dei dati all'interno del database potrebbe portare a una problematica di tipo *information disclosure*, ovvero i dati potrebbero essere soggetti ad un utilizzo volta alla deduzione di informazioni non destinate ad essere reperibili o divulgate. Uno degli attacchi che mira a riprodurre questa problematica è il *Linking attack*, una tipologia di attacco che punta all'ottenimento di informazioni private su individui presenti in una pubblicazione di dati, spesso attraverso lo sfruttamento di altre fonti o risorse in modo da poterle correlare e raggiungere una maggiore conoscenza sui dati presi in esame.

A sua volta, una problematica di Information disclosure si suddivide in tre sotto-problematiche:

- *Identity disclosure*: quando l'attaccante è in grado di risalire all'identità di un rispondente tramite i dati rilasciati nella base di dati.
- *Attribute disclosure*: quando l'attaccante è in grado di attribuire ad un rispondente le informazioni di uno o più attributi rilasciati nella base di dati.
- *Inferential disclosure*: quando l'attaccante è in grado di scoprire correlazioni nei dati, provenienti da fonti diverse, che lo portano ad avere una maggiore conoscenza del rispondente (producono un abbassamento dell'incertezza).

2 K-anonimato

2.1 Protezione dei dati

In questo documento, si vuole affrontare il problema della protezione dei dati degli utenti relativi ai record salvati nei database di microdata. Come prima cosa, è necessario definire l'*Anonimato*: garantire l'anonimato dei dati significa garantire l'anonimato dei soggetti a cui i dati fanno riferimento. In particolare l'obbiettivo principale è permettere il rilascio di informazioni di una determinata base di dati garantendo l'anonimato delle informazioni degli individui presenti nel tabella.

Per evitare problemi legati al rischio di esposizione di un determinato soggetto attraverso un record presente in una base di dati, è fondamentale attuare una limitazione dei dati e dell'accesso alla stessa.

- Per *dati limitati*, si intende l'esclusione dal database di informazioni che permettono una precisa e definita identificazione di un soggetto (ad esempio: numero della carta d'identità, codice fiscale, nome e cognome, indirizzo), in modo che i dati rilasciati siano privi di informazioni specifiche del soggetto.
- Un altro aspetto fondamentale è la *limitazione dell'accesso* ai dati contenuti nella base di dati, ovvero solo alcuni soggetti possono accedervi in determinate condizioni.

Può essere fondamentale anche l'attuazione di pratiche volte a ridurre il rischio di problemi legati all'esposizione dei dati, come ad esempio il rilascio del database dopo un certo periodo di tempo per ottenere un disallineamento temporale tra le diverse informazioni reperibili.

Tuttavia, questi accorgimenti da soli non bastano per garantire l'anonimato dei dati.

Ci sono ulteriori tecniche da attuare per aumentare la protezione dei dati e per ottenere il k-anonimato in una base di dati. Queste tecniche di protezione includono:

- Tecniche di mascheramento, che si suddividono ulteriormente in:
  - Non perturbative (i dati originali non vengono modificati, ma alcuni dati sono nascosti o rimossi tramite generalizzazione o soppressione)
  - Perturbative (i dati originali vengono modificati)
- Rilascio di dati sintetici, che comprende:
  - Completamente sintetici (dati completamente generati)
  - Parzialmente sintetici (dati contenenti una combinazione di dati generati e dati reali)

In particolare, negli algoritmi che cercano di applicare il K-anonimato, sono fondamentali le tecniche di generalizzazione e di soppressione.

- *Generalizzazione*: è una tecnica non perturbativa dei dati, attraverso la quale i valori di uno specifico attributo vengono sostituiti da valori più generalizzati in modo tale da rimuovere i dettagli specifici dei dati risultanti, rendendo così l'identificazione del soggetto molto più difficile.
- *Soppressione*: anche questa è una tecnica non perturbativa dei dati, attraverso la quale alcune informazioni che potrebbero in qualche modo portare all'identificazione del soggetto sono rimosse dai dati rilasciati. È una tecnica che può essere utile in alcuni casi in sostituzione della generalizzazione, specialmente se a causa di uno o pochi record (specialmente outliers) saremmo costretti ad avere una generalizzazione troppo ampia.

Inoltre, possiamo applicare queste tecniche a diversi livelli di granularità all'interno del nostro database: la generalizzazione può essere effettuata a livello di colonna o di singola cella, mentre la soppressione può essere applicata a livello di riga/record, di colonna o di singola cella.

2.2 K-Anonimato

Una tabella garantisce il K-anonimato quando ogni tupla presente può essere relazionato con almeno k altre tuple, con k intero positivo scelto dal proprietario della base di dati.

K-anonimato 1 *Requisito di k-anonimato per una release di dati deve essere tale che ogni combinazione di valori dei quasi-identificatori QI può essere indistinguibilmente corrisposta ad almeno k identità[1].*

Definizione completa:

K-anonimato 2 *Data una tabella $T(A_1, \ldots, A_n)$ e un attributo QI quasi-identificatore associato. T soddisfa la k-anonimità rispetto a QI se e solo se ogni sequenza di valori in $T[QI]$ appare almeno con k occorrenze in $T[QI]$.[9]*

Per comprendere appieno il significato del K-Anonimato, è essenziale comprendere l'importanza della possibilità di collegamento tra diverse fonti e record di basi di dati. Questa possibilità aumenta con il crescere del numero di attributi comuni tra le basi di dati, l'accuratezza e la precisione dei dati e il numero di fonti esterne che contengono dati uguali o similari. Per verificare questa problematica è fondamentale la conoscenza di sorgenti di dati esterne che possono essere utilizzate per effettuare collegamenti sui dati, un'assunzione che nella pratica risulta spesso inattuabile. È quindi importante adottare un approccio che, considerando ogni combinazione di quasi-identificatori, garantisca che i valori presenti nel dataset abbiano almeno k occorrenze.

Possiamo definire tre tipologie di "protezioni" per cercare di anonimizzare il database:

- *K-anonimato*: i dati relativi a un determinato soggetto presenti in una base di dati devono essere indistinguibili rispetto a un dato insieme di attributi; i dati devono quindi avere almeno "k" occorrenze per ogni quasi-identificatore presente. Il k-anonimato è considerato sicuro rispetto ai "linking attack".
- *L-diversità*: proprietà di una base di dati che si definisce valida se, dopo aver applicato il k-anonimato, nei diversi gruppi di attributi sensibili si hanno almeno "l" valori diversi. È una tecnica che rende più difficili "homogeneity attack" e "background knowledge attack".[8]
- *T-chiusura*: indica che la distanza tra le distribuzioni dei valori sensibili presenti nel mondo reale e quelli presenti nella base di dati, dopo aver applicato il k-anonimato, deve essere minore di un dato valore "t". Questo permette la difesa da attacchi di tipo "skewness".[5]

2.2.1 Generalizzazione

Generalizzazione 1 *Date due tabelle T_x e T_y definite sullo stesso insieme di attributi, T_y è una generalizzazione (con eventuale soppressione) della tabella T_x se:*

- $|T_y| \leq |T_x|$ *(data eventuale soppressione)*
- *per ciascun attributo di T_y si ha un dominio uguale al dominio di partenza o una sua generalizzazione*

- *possibile definire f iniettiva che associa a elementi distinti nel dominio elementi distinti nel codominio (non possibile in tutto il codominio per la differenza di cardinalità), in generale $\forall t_y \in T_y, f(t_y) = t_x$, con $t_x \in T_x$ tale che t_y uguale o generalizzazione di t_x*

Generalizzazione sui domini Possiamo definire una relazione di generalizzazione sui domini (solitamente indicata con $\leq_D$, che dati due o più domini in relazione tra loro implica l'esistenza per ogni dominio di una gerarchia di generalizzazione basata sui domini $DGH_D = (Dom, \leq_D)$ La generalizzazione sui domini presenta le seguenti proprietà:

- Proprietà P1: $\forall D_i, D_j, D_k \in Dom$: con $D_i \leq_D D_j, D_i \leq_D D_k \Rightarrow D_j \leq_D D_k \vee D_k \leq_D D_j$;
- Proprietà P2: tutti i massimi elementi del Dominio sono singoli.

Generalizzazione sui valori Possiamo definire una relazione dei valori sui domini (solitamente indicata con $\leq_V$, che dati due domini associa ogni valore del primo dominio con un unico valore del secondo dominio, in questo caso il primo dominio è in diretta generalizzazione con il secondo dominio. Ciò implica l'esistenza per ogni dominio di una gerarchica di generalizzazione basata sui valori $VGH_D = (Dom, \leq_V)$

Il parametro *K-Minimale* permette di ottenere la migliore delle tabelle possibili in base alla necessità dei dati che devono essere rilasciati mantenendo comunque valida il K-Anonimato, in particolare può dare un'indicazione generale sul trade-off tra privatezza e utilità di una base di dati.

K-Minimal 1 *Date due tabelle T_y e T_x tali per cui $T_y \leq T_x$, definiamo $MassimoSuperiore$ come soglia di soppressione accettabile. T_y è una generalizzazione $k-minimal$ della tabella T_x se:*

- *T_y soddisfa $k-anonymity$ con soppressione minima (non sopprime più di quello che si necessiti).*
- *$|T_x| - |T_y| \leq MassimoSuperiore$, la soppressione non è superiore alla soglia massima accettabile.*
- *Non esiste un'ulteriore tabella che soddisfa le prime due condizioni con minore generalizzazione di T_y*

2.2.2 Generalizzazione ottimale

Possiamo derivare dei criteri per poter determinare se una generalizzazione è ottimale per una determinata base di dati. I criteri sono i seguenti:

- Distanza minima assoluta: dato un insieme di dati in cui è stata applicata una generalizzazione, si tende a preferire quella con la distanza minima assoluta (cioè quella con il minor numero di passaggi di generalizzazione svolti).
- Distanza minima relativa: dato un insieme di dati in cui è stata applicata una generalizzazione, si tende a preferire quella con la distanza relativa più piccola (cioè quella con il minore numero di passaggi considerando il numero di passaggi effettuati rispetto ai possibili passaggi di generalizzazione).

- Massima distribuzione: dato un insieme di dati in cui è stata applicata una generalizzazione, si tende a preferire quella che produce il maggior numero di tuple distinte.
- Minima soppressione: dato un insieme di dati in cui è stata applicata una generalizzazione, si tende a preferire quella che comporta il minor numero di tuple soppresse, ovvero quella in cui viene eliminato il minor numero di tuple, di conseguenza quella con la cardinalità maggiore.

2.3 Tipologie di attacchi

Come detto in precedenza, le basi di dati condivise pubblicamente possono essere soggette a diverse tipologie di attacchi da parte di utenti malintenzionati:

- *Linking Attack*: consiste nel tentativo di ri-identificare l'identità di un individuo (presente nella base di dati) combinando i dati con altre informazioni disponibili nel mondo reale. È possibile mitigare questo attacco rendendo la base di dati K-anonima.
- *Homogeneity attack*: attacco si basa sulle informazioni di un determinato soggetto che permettono di categorizzarlo in un macro-gruppo della base di dati per ricavarne informazioni in grado di identificare l'individuo. È possibile mitigare questo attacco rendendo la base di dati K-anonima e utilizzando la proprietà di l-diversità.
- *Background/External Knowledge attack*: attraverso l'utilizzo di informazioni esterne alla base di dati, l'attaccante ha la possibilità di dedurre con una buona probabilità informazioni sull'individuo basandosi sulle informazioni sensibili presenti nella base di dati (ad esempio se viene ricavata l'informazione che un utente vive in una determinata area è possibile ricavare delle informazioni riguardanti la locazione dell'utente nella base di dati, se non accuratamente suddivisa), la divulgazione di informazioni può avvenire:
  - su un utente target specifico.
  - su altri individui del gruppo a cui è possibile ricondurre l'utente target.
  - su un gruppo si soggetti che hanno stessi valori dei dati sensibili dell'utente target.

  È possibile diminuire l'incidenza di questo attacco rendendo la base di dati K-anonima e utilizzando la proprietà di l-diversità.
- *Skewness attack*: attacco che si basa sulla distribuzione dei valori sensibili nei dati rilasciati nella base di dati, se la distribuzione dei valori sensibili nel database è diversa da quella reperibile dal mondo reale (o dalle altre basi di dati che trattano lo stesso argomento) in questo modo si potrebbe avere un rilascio involontario di informazioni in grado di rilevare informazioni di uno o più individui. È possibile mitigare questo attacco rendendo la base di dati K-anonima e utilizzando la proprietà di t-chiusura.
- *Similarity attack*: avviene quando i valori sensibili sono diversi ma semanticamente vicini all'interno della base di dati (ad esempio in una base di dati riguardante il campo medico potrebbero esserci diverse patologie che causano problemi allo stomaco, potrebbe sussistere la problematica che in un gruppo di record pur trattandosi di patologie differenti è possibile ricondurre la patologia di un utente ad una problematica dello stomaco in quel caso si avranno valori dei dati sensibili simili tra

loro), come per skewness attack, si potrebbe quindi avere un rilascio involontario di informazioni. È possibile prevenire questa tipologia di attacchi rendendo la base di dati K-anonima e attraverso la proprietà di l-diversità.

- *Rilasci multipli (rilasci longitudinali)*: alcune basi di dati potrebbero necessitare di frequenti modifiche riguardanti alcuni dati di determinati utenti, che richiedono regolari pubblicazioni, un utente malintenzionato potrebbe provare a comparare le varie differenze nei dati pubblicati e di conseguenza correlare i dati rilasciati di volta per volta. È importante prevenire rilasci non controllati di informazioni utilizzando la proprietà di m-invarianza (proprietà che implica la capacità di una base di dati di mantenere la sua coerenza e validità dei dati nonostante eventuali cambiamenti nelle condizioni o nei dati rilasciati).

3 Algoritmi

3.1 Algoritmi per il K-anonimato

Per garantire il k-anonimato minimo in una tabella estratta da una base di dati, è necessario applicare diverse operazioni di generalizzazione e soppressione delle tuple della tabella stessa. Questo problema rientra tra i problemi NP-hard, pertanto la maggior parte degli algoritmi esistenti presenta un tempo computazionale esponenziale che cresce con il numero di attributi che compongono i quasi-identificatori.

3.1.1 Tecniche per il K-anonimato

Esistono diversi approccio per la gestione del K-anonimato in una base di dati e diverse modi per gestire generalizzazione e soppressione con i vari algoritmi.

Come visto in precedenza la generalizzazione può essere applicata a livello di attributo (AG) e a livello di cella (CG), mentre la soppressione può essere applicata a livello di tupla (TS), a livello di attributo (AS) e a livello di cella (AS), ciò produce diverse possibili combinazioni attraverso le quali applicare algoritmi di generalizzazione (con soppressione) per poter raggiungere l'obbiettivo del k-anonimato:

	Soppressione			
Generalizzazione	Tuple	Attributi	Celle	None
Attributi	AG_TS	AG_AS	AG_CS	AG_
Celle	CG_TS	CG_AS	CG_CS	CG_
None	_TS	_AS	_CS	-

Table 1: tecniche di k-anonimato[1]

In particolare analizzandole:

- *AG_TS*: generalizzazione applicata a livello di attributo (colonna) e soppressione a livello di tupla (riga), è l'assunzione generale del k-anonimato e viene utilizzata nella maggior parte degli algoritmi principali poiché ha un ottimo trade-off tra qualità dei dati e complessità computazionale;
- *AG_AS*: sia generalizzazione sia soppressione applicate a livello di attributo (colonna), solitamente non utilizzata, inoltre in quanto se viene effettuata la generalizzazione a livello di attributo non vi è la necessita di utilizzare anche la soppressione ciò la rende equivalente al modello *AG_*;
- *AG_CS*: generalizzazione applicata a livello di attributo (colonna) e soppressione a livello di singola cella, tecnica che permette di ridurre gli effetti della soppressione, in quanto al posto di agire sull'intera tupla agisce solamente sulla cella da sopprimere, ha maggiore complessità a livello computazionale e perciò è poco utilizzata, ne fanno uso gli algoritmi: Datafly software e μ-argus (applicano gli stessi principi del k-anonimato ma senza la garanzia di soluzioni minimali;
- *CG_CS*: sia generalizzazione sia soppressione applicate a livello di cella, approccio che in generale può essere inteso come generalizzazione al massimo elemento della gerarchia per quella data cella, in modo analogo ad AG_AS $\Rightarrow$ AG_ anche a livello di cella se viene effettuata la generalizzazione a livello di celle non vi è la necessità di utilizzare anche la soppressione, ciò la rende equivalente al modello *CG_*;
- *_TS*: generalizzazione non applicata mentre la soppressione applicata a livello di tupla (riga) (può essere ricondotta al modello AG_TS nel quale la gerarchia di generalizzazione ha altezza 0, ha quindi applicabilità molto limitata (casi speciali);

- *_AS*: generalizzazione non applicata mentre la soppressione applicata a livello di attributo (colonna), in modo analogo a _TS può essere ricondotta al modello AG_AS nel caso in cui la gerarchia di generalizzazione ha altezza 1;
- *_CS*: generalizzazione non applicata mentre la soppressione applicata a livello di cella, anche questa può essere ricondotta al modello AG_CS nel caso in cui la gerarchia di generalizzazione ha altezza 1;
- *CG_TS*: generalizzazione applicata a livello di cella e soppressione applicata a livello di tupla (riga) non applicabile siccome CG porta a generalizzare al massimo elemento della gerarchia di generalizzazione che implica quindi la soppressione allo stesso livello;
- *CG_AS*: generalizzazione applicata a livello di cella e soppressione applicata a livello di tupla (riga) non applicabile siccome CG porta a generalizzare al massimo elemento della gerarchia di generalizzazione che implica quindi la soppressione allo stesso livello.

Tra tutte le possibilità quindi solamente poco sono effettivamente utilizzabili per costruire algoritmi per gestire generalizzazione e soppressione, in particolare sono:

1. *AG_TS* (la più estensivamente studiata)
2. *AG_* (che fa riferimento a AG_,AG_AS,_AS)
3. *CG_* (che fa riferimento a CG_,CG_CS)
4. *AG_CS* (che fa riferimento a AG_CS, _CS)
5. *_TS* (applicabilità limitata a casi particolari)

3.2 Rinforzo basato sulla generalizzazione

La generalizzazione (con eventuale applicazione della soppressione per gli elementi outliers) può essere applicata a diversi livelli di granularità:

Generalizzazione

- livello di cella (sostituendo il valore della cella con valori più generali)
- livello di attributo (generalizzando il valore dell'attributo di ogni cella nella colonna)

Soppressione

- livello di cella (rimozione della singola cella)
- livello di attributo (rimozione della colonna corrispondente all'attributo)
- livello di record (rimozione di una tupla dalla tabella)

Oltre al livello di granularità in cui vengono applicati questi approcci, è possibile definire due famiglie di tecniche:

1. Generalizzazione basata sulla gerarchia
2. Generalizzazione basata sui record

3.2.1 Generalizzazione basata sulla gerarchia

Questo tipo di generalizzazione è basato sulla definizione di una gerarchia di generalizzazione per ogni attributo quasi-identificatore della tabella. Per questa tipologia di generalizzazione vengono utilizzate due tipi di metriche:

- *elemento massimale*: il valore più generale che può essere definito per un determinato attributo.
- *elementi minimali*: i valori più specifici di un determinato attributo (i valori che appaiono originalmente nella base di dati).

In generale, un dominio presente nella gerarchia della generalizzazione per ogni attributo QI contiene il suo valore usato per poterlo generalizzare. La precedente definizione di relazione di generalizzazione ($\leq_D$) implica l'esistenza di una gerarchia totalmente ordinata per ogni dominio presente.

Il problema generale del calcolo del K-anonimato minimizzando la generalizzazione utilizzandone la gerarchia è NP-Hard. Per questo motivo sono stati proposti sia algoritmi esatti che euristici. Due esempi di algoritmi esatti che verranno discussi successivamente sono:

- *K-Minimal*, Samarati: utilizza la ricerca binaria sulla gerarchia di generalizzazioni per evitare di cercare nell'intero spazio di generalizzazione. Inoltre, utilizza i distance vector per poter calcolare la distanza tra domini generalizzati, in modo da verificare che il requisito di K-anonimato venga rispettato.
- *Incognito*, LeFevre: si basa sull'osservazione secondo la quale se un dataset è K-anonimo per un set di QI, allora sarà anche K-anonimo per ogni sottoinsieme dei QI (condizione necessaria ma non sufficiente). Questo permette di calcolare in modo efficiente il K-anonimato di un dataset minimizzando l'adozione delle generalizzazioni e di conseguenza scartando le generalizzazioni per i quali i QI non soddisfano il K-anonimato.

3.2.2 Generalizzazione basata sui record

Questo tipo di generalizzazione si basa sulla generalizzazione del valore dei quasi-identificatori in intervalli. Solitamente viene utilizzata quando il controllo del K-anonimato viene effettuato a run-time e non richiede la pre-definizione di gerarchie di generalizzazione. Tuttavia, questo approccio richiede un ordinamento dei valori degli attributi quasi-identificatori (QI), uno degli algoritmi più utilizzati per questa tipologia di rinforzo è l'algoritmo *Mondrian*, che generalizza gli attributi a livello di singola cella. Questo algoritmo utilizza una rappresentazione spaziale dei dati e opera in modo ricorsivo, partizionando lo spazio in regioni o frammenti contenenti un insieme di punti che corrispondono a record della tabella. L'algoritmo termina quando ogni successivo partizionamento genererebbe una regione con meno di k punti. I valori dei QI risultanti nel frammento vengono quindi sostituiti a livello delle tuple rispetto ai valori originali.

3.3 Rinforzo basato sulla frammentazione e microaggregazione

Esistono ulteriori tecniche per garantire la protezione dei dati degli utenti nelle basi di dati, ovvero l'adozione della frammentazione dei dati e della microaggregazione.

3.3.1 Frammentazione dei dati

Questo approccio è un'alternativa alla generalizzazione e, come la generalizzazione, produce dati veritieri, in particolare consiste nella suddivisione "verticale" del dataset originale in frammenti, in modo da nascondere la corrispondenza specifica tra quasi-identificatori e informazioni sensibili. Sono state proposte diverse metodologie per eseguire lo splitting, tra cui *Anatomy*, che permette di produrre basi di dati l-diverse.

1. La prima operazione raggruppa le tuple originali de-identificate in gruppi contenenti almeno l valori sensibili.
2. I gruppi sono successivamente suddivisi per separare i quasi-identificatori dagli attributi sensibili, i quali sono conteggiati tramite un attributo *count* che ne indica le occorrenze nel gruppo. Solitamente è possibile ricostruire il frammento originale complementando tramite un attributo ID, a livello di granularità di gruppo.

3.3.2 Microaggregazione

Un altro approccio alternativo per raggiungere la proprietà di k-anonimato, con esito perturbativo della base di dati, è la microaggregazione. La quale può essere definita come una serie di operazioni di partizionamento seguite da operazioni di aggregazione.

- Partizionamenti: simili alle operazioni di frammentazione, i set di tuple sono partizionati in cluster diversi in modo che:
  - le tuple nello stesso cluster siano simili tra loro;
  - ogni cluster contenga almeno k elementi.
- Aggregazioni: viene computato un operatore di aggregazione (solitamente la media per dati numerici e la mediana per dati categorici) per ogni cluster e per ogni attributo quasi-identificatore. Il risultato calcolato viene sostituito al valore originale di tutti i quasi-identificatori del cluster.

3.4 Tecniche per il K-anonimato

Esistono diversi approcci per la gestione del k-anonimato in una base di dati e vari modi per implementare la generalizzazione e la soppressione tramite differenti algoritmi.

Come discusso in precedenza, la generalizzazione può essere applicata a livello di attributo (AG) o a livello di cella (CG), mentre la soppressione può essere applicata a livello di tupla (TS), di attributo (AS) e di cella (CS). Questo produce diverse combinazioni attraverso le quali si possono applicare algoritmi di generalizzazione (con soppressione) per raggiungere l'obiettivo del k-anonimato:

	Soppressione			
Generalizzazione	Tuple	Attributi	Celle	None
Attributi	AG_TS	AG_AS	AG_CS	AG_
Celle	CG_TS	CG_AS	CG_CS	CG_
None	_TS	_AS	_CS	-

Table 2: Tecniche di k-anonimato [1]

Di seguito un'analisi dettagliata delle combinazioni:

- *AG_TS*: generalizzazione applicata a livello di attributo (colonna) e soppressione a livello di tupla (riga). È l'assunzione generale del k-anonimato ed è utilizzata

nella maggior parte degli algoritmi principali poiché offre un ottimo trade-off tra qualità dei dati e complessità computazionale.

- AG_AS: sia la generalizzazione che la soppressione sono applicate a livello di attributo (colonna). Solitamente non è utilizzata poiché, se viene effettuata la generalizzazione a livello di attributo, non vi è necessità di utilizzare anche la soppressione, rendendola equivalente al modello $AG\_$ che è a sua volta sussunto a AG_TS.
- AG_CS: generalizzazione applicata a livello di attributo (colonna) e soppressione a livello di singola cella. Questa tecnica riduce gli effetti della soppressione, poiché agisce solo sulla cella da sopprimere anziché sull'intera tupla. Tuttavia, ha una maggiore complessità computazionale e quindi è meno utilizzata, sebbene venga impiegata da algoritmi come Datafly e μ-Argus (che applicano principi simili al k-anonimato ma senza garanzia di soluzioni minimali).
- CG_CS: sia la generalizzazione che la soppressione sono applicate a livello di cella. Questo approccio può essere visto come una generalizzazione al massimo livello della gerarchia per quella cella, analogamente a AG_AS è sussunto ad $AG\_$ e di conseguenza ad AG_TS. Se viene effettuata la generalizzazione a livello di cella, non vi è necessità di soppressione, rendendola equivalente al modello $CG\_$.
- $\_TS$: generalizzazione non applicata, mentre la soppressione è applicata a livello di tupla (riga). Questo caso può essere ricondotto al modello AG_TS con una gerarchia di generalizzazione di altezza zero, limitandone l'applicabilità a casi speciali.
- $\_AS$: generalizzazione non applicata, mentre la soppressione è applicata a livello di attributo (colonna). Analogamente a $\_TS$, può essere ricondotta al modello AG_AS nel caso in cui la gerarchia di generalizzazione abbia altezza uno.
- $\_CS$: generalizzazione non applicata, mentre la soppressione è applicata a livello di cella. Anche questo caso può essere ricondotto al modello AG_CS con una gerarchia di generalizzazione di altezza uno.
- CG_TS: generalizzazione applicata a livello di cella e soppressione a livello di tupla (riga). Non è applicabile poiché CG porta a generalizzare al massimo elemento della gerarchia di generalizzazione, che implica la soppressione allo stesso livello.
- CG_AS: generalizzazione applicata a livello di cella e soppressione a livello di attributo (colonna). Non è applicabile per lo stesso motivo di CG_TS.

Tra tutte le combinazioni possibili, solo alcune sono effettivamente utilizzabili per costruire algoritmi per la gestione della generalizzazione e della soppressione. In particolare, queste sono:

1. AG_TS (la più estensivamente studiata)
2. $AG\_$ (che include $AG\_$, AG_AS, $\_AS$)
3. $CG\_$ (che include $CG\_$, CG_CS)
4. AG_CS (che include AG_CS, $\_CS$)
5. $\_TS$ (applicabilità limitata a casi particolari)

3.5 Algoritmi per il k-anonimato

Come visto per le tecniche di generalizzazione e soppressionesection 3.4, anche gli algoritmi per il k-anonimato possono essere suddivisi in varie classi in base al loro approccio alla generalizzazione e alla soppressione.

3.5.1 Algoritmi basati su AG_TS e AG_

- AG_TS: generalizzazione sugli attributi e soppressione delle tuple.
- $AG\_$: generalizzazione sugli attributi.

All'interno di queste classi di generalizzazione, è possibile individuare diversi algoritmi.

Algoritmo di ricerca binaria per K-Minimal (Samarati) Questo algoritmo fa affidamento alla generazione di una soluzione basata sulla definizione di k-minimal. Viene creato un albero con le soluzioni della generalizzazione, dove ogni percorso nella gerarchia dei domini rappresenta una generalizzazione della tabella originale. Il nodo più basso di ogni percorso, che soddisfa il k-anonimato, è definito generalizzazione minima locale. Questa costruzione garantisce che ogni generalizzazione k-minimale sia localmente minima rispetto a un determinato percorso. Il funzionamento dell'algoritmo si basa sulla valutazione delle soluzioni ad altezze specifiche della gerarchia e continua fino a quando non viene raggiunta l'altezza più bassa per la quale esiste un vettore di distanza che soddisfa il k-anonimato.

1. Vengono valutate le soluzioni ad altezza $\frac{h}{2}$ della gerarchia.
2. Se esiste almeno una soluzione che rispetta i vincoli imposti e soddisfa il k-anonimato, si procede valutando le soluzioni ad altezza $\frac{h}{4}$ della gerarchia. Altrimenti, si valuta la soluzione all'altezza $\frac{3h}{4}$, poiché le soluzioni sottostanti nel cammino sono meno generalizzate e quindi non possono essere soluzioni k-minimali.
3. L'algoritmo prosegue in questo modo fino a quando non viene raggiunta l'altezza più bassa per la quale esiste un vettore di distanza che soddisfa il k-anonimato.

Questo approccio permette di trovare una possibile soluzione, ma non garantisce la ricerca della soluzione ottimale.

Algoritmo k-optimize L'algoritmo k-optimize si basa su tre concetti fondamentali:

- Ordinamento degli attributi e dei valori nel dominio di appartenenza: cruciale evidenziare che l'ordine scelto per gli attributi e i valori nel dominio influenza il processo di generalizzazione.
- Associazione di un valore (intero) come indice a ogni valore presente nel dominio, seguendo l'ordinamento definito in precedenza.
- Generalizzazione interpretata come unione dei singoli valori degli indici, dove il valore minimo in un dominio viene omesso. Ad esempio, vengono raggruppati tutti gli elementi prima di un determinato indice e tutti gli elementi dopo un determinato indice.

Il funzionamento dell'algoritmo si basa sulla visita dell'albero di generalizzazione: partendo dal nodo radice, che rappresenta anche la soluzione possibile del problema ($\emptyset$), inteso come massima generalizzazione, l'algoritmo discende lungo l'albero visitando tutti i nodi ai quali sono associati dei costi, che riflettono la quantità di generalizzazione e soppressione necessaria per anonimizzare la tabella (ovvero rappresentano la perdita di informazioni per quella determinata soluzione). La visita procede finché non incontra un nodo che non è una soluzione. A questo punto, poiché la generalizzazione diminuisce, gli altri nodi presenti non possono essere soluzione e viene quindi effettuato il "pruning" (taglio) di quel ramo.

Algoritmo Incognito L'Algoritmo Incognito si basa sulla necessità di stabilire un valore k di occorrenze identiche per ciascun attributo e combinazione di quasi-identificatori, al fine di garantire il soddisfacimento del k-anonimato. Appunto importante è che questa condizione è necessaria, ma non è sufficiente, poiché potrebbe sussistere il caso in cui tutte le combinazioni siano diverse e quindi si otterrebbe $k = 1$.

Il funzionamento dell'algoritmo avviene attraverso una serie di iterazioni:

1. **Iterazione 1**: Viene verificata la condizione necessaria, precedentemente definita, per garantire il k-anonimato. Durante questa fase, possono essere scartate tutte le soluzioni (generalizzazioni) che non soddisfano tale condizione.
2. **Iterazione 2**: Vengono combinate tutte le generalizzazioni rimanenti a coppie e viene effettuato il controllo della condizione per ciascuna coppia ottenuta, eventualmente scartando le soluzione che non la rispettano?.
3. ...
4. **Iterazione n**: Viene restituito il risultato finale ottenuto dall'algoritmo.

L'algoritmo termina una volta raggiunto il passo n, corrispondente al numero di quasi-identificatori presenti nella tabella selezionata.

Algoritmi Euristici Gli algoritmi discussi in precedenza mirano a trovare la soluzione esatta per il problema del k-anonimato. Tuttavia, essendo questo problema riconducibile alla categoria dei problemi NP-Hard, tali algoritmi presentano una complessità esponenziale. Per questo motivo, sono stati sviluppati approcci alternativi, come gli algoritmi euristici, che risolvono l'anonimato utilizzando diverse metodologie:

- Iyengar utilizza algoritmi genetici e un metodo di ricerca incompleto stocastico.
- Winkler impiega una tecnica di annealing simulata per trovare una soluzione minima locale.
- Fung, Wang e Yu adottano un approccio euristico top-down che utilizza sia attributi continui che categorici. Questo metodo parte dalla soluzione più generale e successivamente specializza i valori della soluzione corrente fino a raggiungere un punto in cui il k-anonimato viene violato.

A causa della loro natura, gli algoritmi euristici non possono garantire la qualità ottimale della soluzione. Tuttavia, essi risultano validi quando i limiti computazionali rendono impossibile l'utilizzo di algoritmi che cercano una soluzione esatta.

3.5.2 Algoritmi basati su _CS e CG_

- *_CS*: soppressione delle celle.
- *CG_*: generalizzazione delle celle.

Anche all'interno di queste classi di generalizzazione, è possibile individuare diversi algoritmi.

Algoritmo Mondrian Multidimensionale L'Algoritmo Mondrian Multidimensionale si basa sul concetto di multi-dimensionalità degli oggetti o record salvati nella tabella. Ogni attributo quasi-identificatore rappresenta una dimensione e ogni tupla rappresenta un punto nello spazio definito dai quasi-identificatori. Quando si trovano tuple con lo stesso valore per i quasi-identificatori, vengono rappresentate come punti nello spazio, associati a un valore che ne indica il numero di occorrenze che ne sono state trovate, valori attraverso i quali vengono ordinati lungo gli assi.

Una volta rappresentate tutte le tuple nello spazio multidimensionale, questo viene suddiviso in modo tale che ogni area contenga almeno k occorrenze, garantendo così il rispetto del vincolo per il k-anonimato. Ottenuta la suddivisione, tutti i punti che si trovano nello stesso "cluster" vengono generalizzati a un valore univoco, che rappresenta la loro generalizzazione. I valori dei quasi-identificatori delle tuple corrispondenti vengono quindi sostituiti con il valore della generalizzazione associata.

L'algoritmo è completo e flessibile in quanto può operare in diverse modalità:

- in singola o multi-dimensione, in base al numero di attributi;
- con diverse strategie di generalizzazione, ad esempio globale sugli attributi o locale a livello di cella;
- con diverse strategie di suddivisione, come relax o strict;
- con diverse metriche per determinare la suddivisione su ogni dimensione.

Algoritmi di approssimazione Per risolvere generalizzazione del tipo $\_CS$ e $CG\_$ sono stati proposti algoritmi di approssimazione, garantiscono che il costo di una soluzione C rispetto al costo della soluzione ottimale C^* sia minore di un p-approssimazione ovvero: $\frac{C}{C^*} \leq p$

Algoritmi di Approssimazione Per affrontare la generalizzazione dei tipi $\_CS$ e $CG\_$, sono stati sviluppati anche algoritmi di approssimazione. Questi algoritmi sono progettati per fornire soluzioni efficienti in termini di tempo di calcolo, garantendo che il costo di una soluzione C rispetto al costo della soluzione ottimale C^* sia entro un fattore di p-approssimazione, ossia: $\frac{C}{C^*} \leq p$.

3.5.3 Algoritmi Distribuiti

È essenziale anonimizzare anche i dati distribuiti tra nodi interconnessi. Per questo motivo, sono stati sviluppati algoritmi per garantire il k-anonimato in ambito distribuito.

- Due gruppi di studio separati, Jiang e Clifton [4] e Wang, Fung e Dong [12], hanno proposto protocolli di comunicazione per riunire le tabelle (precedentemente partizionate verticalmente) in un unico dataset. Questo viene ottenuto tramite un protocollo che decide quali tuple generalizzare in base ai quasi-identificatori, determinati durante l'interazione tra le parti.
- Zhong, Yang e Wright [13] hanno proposto un algoritmo di k-anonimato distribuito per tabelle partizionate orizzontalmente. Questo algoritmo prevede la crittografia degli attributi sensibili tramite una chiave di codifica. La decrittografia è possibile solo se ci sono almeno k tuple con lo stesso valore del corrispondente quasi-identificatore, una tecnica che corrisponde alla soppressione delle tuple.

4 Analisi Applicativa

4.1 Test Applicativo degli Algoritmi

Per testare gli algoritmi per garantire il k-anonimato sui database, sono stati adottati due approcci distinti, basati su un progetto di Longtao Zheng[6], per il testing applicativo degli algoritmi:

- Per gli algoritmi basati su AG_TS e AG_, si è optato per l'utilizzo dell'algoritmo proposto da Samarati.
- Per gli algoritmi basati su _CS e CG_, la scelta è ricaduta sull'algoritmo proposto da Mondrian.

Il test a livello pratico è finalizzato a valutare l'efficacia degli algoritmi di Samarati e Mondrian nel garantire il k-anonimato, mantenendo nel contempo la qualità e l'utilità dei dati. Si prevede di misurare i tempi di esecuzione e la perdita di informazioni causata dalle tecniche di anonimizzazione, al fine di valutarne l'efficienza in diversi scenari, considerando i dataset Adult e Bank.

4.2 Dataset Utilizzati

Per la sperimentazione degli algoritmi di anonimizzazione, sono stati utilizzati due dataset principali:

- **Adult Dataset** [10]: Questo dataset, reperibile presso l'archivio dell'UCI Machine Learning Repository è uno dei dataset di benchmark più comunemente utilizzati per testare algoritmi di anonimizzazione, include attributi quali 'age', 'work_class', 'final_weight', 'education', 'education_num', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country' e 'class'.
- **Bank Marketing Dataset** [11]: Anch'esso disponibile presso l'archivio UCI, contiene attributi come 'age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome' e 'y'.

In aggiunta per i dataset utilizzati nel progetto, sono state definite delle gerarchie di generalizzazione per gli attributi quasi-identificatori, necessarie per l'implementazione degli algoritmi di k-anonimato (Samarati e Mondrian).

Queste gerarchie sono:

- per il dataset Adult:
  - adult_race
    - Other,*
    - Amer-Indian-Eskimo,*
    - Black,*
    - White,*
    - Asian-Pac-Islander,*
  - adult_marital_status
    - NM,*
    - Married,*

        * leave,*
        * alone,*
        * Never-married,NM
        * Married-civ-spouse,Married
        * Married-AF-spouse,Married
        * Divorced,leave
        * Separated,leave
        * Widowed,alone
        * Married-spouse-absent,alone
    - adult_gender
        * Female,*
        * Male,*

- per il dataset Bank:
    - bank_job
        * white-collar, employed
        * blue-collar, employed
        * self-employed,employed
        * admin., white-collar
        * unknown, other
        * unemployed, other
        * management, white-collar
        * housemaid, blue-collar
        * entrepreneur, self-employed
        * student, other
        * blue-collar, blue-collar
        * self-employed, self-employed
        * retired, other
        * technician, blue-collar
        * services, blue-collar

4.3 Algoritmi di Anonimizzazione

Per testare l'anonimizzazione dei dati presenti nei set di dati utilizzati sono stati implementati e valutati due algoritmi in grado di effettuare operazioni sui dati in modo che venga implementato il k-anonimato nei dataset:

4.3.1 Algoritmi

L'algoritmo di Samarati [3] è stato implementato nei file `samarati.py` e `samarati_bank.py` section 3.5.1.

L'algoritmo Mondrian [2] è stato implementato nei file `mondrian.py` e `mondrian_bank.py` section 3.5.2.

4.3.2 Parametri di Esecuzione

Per ottenere risultati ottimali, sono stati definiti i seguenti parametri di esecuzione per ciascun dataset:

Dataset Adult

- **Samarati:** `k = 10`, `maxsup = 20`

```
python main.py --samarati --k 10 --maxsup 20
```

- **Mondrian:** `k = 10`

```
python main.py --mondrian --k 10
```

Dataset Bank

- **Samarati:** `k = 3`

```
python main.py --samarati --k 3
```

- **Mondrian:** `k = 3`

```
python main.py --mondrian --k 3
```

Le configurazioni specifiche per ciascun dataset sono descritte nei file, contenuti nel progetto Github [7] `default_data_config` e `default_data_configB`, situati rispettivamente in `utils/data_loader.py` e `utils/data_loader_bank.py`. I risultati delle esecuzioni sono salvati nella directory `results`.

4.4 Struttura del Programma

Il programma è organizzato nelle seguente albero:

- **algorithms:**
  - `samarati.py`
  - `mondrian.py`
  - `samarati_bank.py`
  - `mondrian_bank.py`
- **data:** contiene i dataset e le gerarchie di generalizzazione.
  - `adult.data`: dataset Adult sorgente.
  - `bank.csv`: dataset Bank sorgente.
  - `gerarchie adult_*`: gerarchie del dataset Adult per i vati QI.
  - `gerarchie bank_*`: gerarchie del dataset Bank per i vari QI.
- **results:** directory dove vengono salvate le tabelle anonimizzate e la tabella originaria.
  - `mondrian.csv`: risultati dataset Adult con Mondrian.
  - `samarati.csv`: risultati dataset Adult con Samarati.
  - `mondrian_bank.csv`: risultati dataset Bank con Mondrian.

  - `samarati_bank.csv`: risultati dataset Bank con Samarati.
  - `original.csv`: tabella originale.

- **utils:**
  - `data_loader.py`: caricatore dati per il dataset Adult.
  - `data_loader_bank.py`: caricatore dati per il dataset Bank.
  - `loss_metrics_adult.py`: metriche di perdita per il dataset Adult.
  - `loss_metrics_bank.py`: metriche di perdita per il dataset Bank.
- `main.py`
- `plot.py`

4.5 Esecuzione degli algoritmi

Per testare l'anonimizzazione dei dati nei due dataset utilizzati, sono stati implementati e valutati due algoritmi: Samarati e Mondrian. Questi algoritmi posso essere eseguiti con parametri specifici e ottimali per ogni dataset, come descritto di seguito.

4.5.1 Esempi di Esecuzione

Samarati su Adult Dataset

```
python main.py --samarati --k 10 --maxsup 20


configuration:
 {'k': 10, 'maxsup': 20, 'samarati': True, 'mondrian':
  ↪  False, 'bank': False, 'data': {'path':
  ↪  'data/adult.data', 'samarati_quasi_id': ['age',
  ↪  'gender', 'race', 'marital_status'],
  ↪  'mondrian_quasi_id': ['age', 'education_num'],
  ↪  'sensitive': 'occupation', 'columns': ['age',
  ↪  'work_class', 'final_weight', 'education',
  ↪  'education_num', 'marital_status', 'occupation',
  ↪  'relationship', 'race', 'gender', 'capital_gain',
  ↪  'capital_loss', 'hours_per_week', 'native_country',
  ↪  'class'], 'samarati_generalization_type': {'age':
  ↪  'range', 'gender': 'categorical', 'race':
  ↪  'categorical', 'marital_status': 'categorical'},
  ↪  'hierarchies': {'age': None, 'gender':
  ↪  'data/adult_gender.txt', 'race': 'data/adult_race.txt',
  ↪  'marital_status': 'data/adult_marital_status.txt'},
  ↪  'mondrian_generalization_type': {'age': 'numerical',
  ↪  'education_num': 'numerical'}}}

row count before sanitizing: 32561
row count sanitized: 30162

hierarchies:
```

```
 {'age': {'39': '(35, 40)', '50': '(50, 55)', '38': '(35,
 ↪  40)', '53': '(50, 55)', '28': '(25, 30)', '37': '(35,
 ↪  40)', '49': '(45, 50)', '52': '(50, 55)', '31': '(30,
 ↪  35)', '42': '(40, 45)', '30': '(30, 35)', '23': '(20,
 ↪  25)', '32': '(30, 35)', '34': '(30, 35)', '25': '(25,
 ↪  30)', '43': '(40, 45)', '40': '(40, 45)', '54': '(50,
 ↪  55)', '35': '(35, 40)', '59': '(55, 60)', '56': '(55,
 ↪  60)', '19': '(15, 20)', '20': '(20, 25)', '45': '(45,
 ↪  50)', '22': '(20, 25)', '48': '(45, 50)', '21': '(20,
 ↪  25)', '24': '(20, 25)', '57': '(55, 60)', '44': '(40,
 ↪  45)', '41': '(40, 45)', '29': '(25, 30)', '47': '(45,
 ↪  50)', '46': '(45, 50)', '36': '(35, 40)', '79': '(75,
 ↪  80)', '27': '(25, 30)', '18': '(15, 20)', '33': '(30,
 ↪  35)', '76': '(75, 80)', '55': '(55, 60)', '61': '(60,
 ↪  65)', '70': '(70, 75)', '64': '(60, 65)', '71': '(70,
 ↪  75)', '66': '(65, 70)', '51': '(50, 55)', '58': '(55,
 ↪  60)', '26': '(25, 30)', '17': '(15, 20)', '60': '(60,
 ↪  65)', '90': '(90, 95)', '75': '(75, 80)', '65': '(65,
 ↪  70)', '77': '(75, 80)', '62': '(60, 65)', '63': '(60,
 ↪  65)', '67': '(65, 70)', '74': '(70, 75)', '72': '(70,
 ↪  75)', '69': '(65, 70)', '68': '(65, 70)', '73': '(70,
 ↪  75)', '81': '(80, 85)', '78': '(75, 80)', '88': '(85,
 ↪  90)', '80': '(80, 85)', '84': '(80, 85)', '83': '(80,
 ↪  85)', '85': '(85, 90)', '82': '(80, 85)', '86': '(85,
 ↪  90)', '(35, 40)': '(30, 40)', '(50, 55)': '(50, 60)',
 ↪  '(25, 30)': '(20, 30)', '(45, 50)': '(40, 50)', '(30,
 ↪  35)': '(30, 40)', '(40, 45)': '(40, 50)', '(20, 25)':
 ↪  '(20, 30)', '(55, 60)': '(50, 60)', '(15, 20)': '(10,
 ↪  20)', '(75, 80)': '(70, 80)', '(60, 65)': '(60, 70)',
 ↪  '(70, 75)': '(70, 80)', '(65, 70)': '(60, 70)', '(90,
 ↪  95)': '(90, 100)', '(80, 85)': '(80, 90)', '(85, 90)':
 ↪  '(80, 90)', '(30, 40)': '(20, 40)', '(50, 60)': '(40,
 ↪  60)', '(20, 30)': '(20, 40)', '(40, 50)': '(40, 60)',
 ↪  '(10, 20)': '(0, 20)', '(70, 80)': '(60, 80)', '(60,
 ↪  70)': '(60, 80)', '(90, 100)': '(80, 100)', '(80, 90)':
 ↪  '(80, 100)', '(20, 40)': '*', '(40, 60)': '*', '(0,
 ↪  20)': '*', '(60, 80)': '*', '(80, 100)': '*'},
 ↪  'gender': {'Female': '*', 'Male': '*'}, 'race':
 ↪  {'Other': '*', 'Amer-Indian-Eskimo': '*', 'Black': '*',
 ↪  'White': '*', 'Asian-Pac-Islander': '*'},
 ↪  'marital_status': {'NM': '*', 'Married': '*', 'leave':
 ↪  '*', 'alone': '*', 'Never-married': 'NM',
 ↪  'Married-civ-spouse': 'Married', 'Married-AF-spouse':
 ↪  'Married', 'Divorced': 'leave', 'Separated': 'leave',
 ↪  'Widowed': 'alone', 'Married-spouse-absent': 'alone'}}

hierarchy heights:
 {'age': 4, 'gender': 1, 'race': 1, 'marital_status': 2}

leaves_num:
```

```
{'age': {'(35, 40)': 5, '(50, 55)': 5, '(25, 30)': 5, '(45,
↪  50)': 5, '(30, 35)': 5, '(40, 45)': 5, '(20, 25)': 5,
↪  '(55, 60)': 5, '(15, 20)': 3, '(75, 80)': 5, '(60,
↪  65)': 5, '(70, 75)': 5, '(65, 70)': 5, '(90, 95)': 1,
↪  '(80, 85)': 5, '(85, 90)': 3, '(30, 40)': 10, '(50,
↪  60)': 10, '(20, 30)': 10, '(40, 50)': 10, '(10, 20)':
↪  3, '(70, 80)': 10, '(60, 70)': 10, '(90, 100)': 1,
↪  '(80, 90)': 8, '(20, 40)': 20, '(40, 60)': 20, '(0,
↪  20)': 3, '(60, 80)': 20, '(80, 100)': 9, '*': 72},
↪  'gender': {'*': 2}, 'race': {'*': 5}, 'marital_status':
↪  {'*': 7, 'NM': 1, 'Married': 2, 'leave': 2, 'alone':
↪  2}}

loss_metric_map:
```

```
{'age': {'*': 1, '39': 0, '50': 0, '38': 0, '53': 0, '28':
↪  0, '37': 0, '49': 0, '52': 0, '31': 0, '42': 0, '30':
↪  0, '23': 0, '32': 0, '34': 0, '25': 0, '43': 0, '40':
↪  0, '54': 0, '35': 0, '59': 0, '56': 0, '19': 0, '20':
↪  0, '45': 0, '22': 0, '48': 0, '21': 0, '24': 0, '57':
↪  0, '44': 0, '41': 0, '29': 0, '47': 0, '46': 0, '36':
↪  0, '79': 0, '27': 0, '18': 0, '33': 0, '76': 0, '55':
↪  0, '61': 0, '70': 0, '64': 0, '71': 0, '66': 0, '51':
↪  0, '58': 0, '26': 0, '17': 0, '60': 0, '90': 0, '75':
↪  0, '65': 0, '77': 0, '62': 0, '63': 0, '67': 0, '74':
↪  0, '72': 0, '69': 0, '68': 0, '73': 0, '81': 0, '78':
↪  0, '88': 0, '80': 0, '84': 0, '83': 0, '85': 0, '82':
↪  0, '86': 0, '(35, 40)': 0.056338028169014086, '(50,
↪  55)': 0.056338028169014086, '(25, 30)':
↪  0.056338028169014086, '(45, 50)': 0.056338028169014086,
↪  '(30, 35)': 0.056338028169014086, '(40, 45)':
↪  0.056338028169014086, '(20, 25)': 0.056338028169014086,
↪  '(55, 60)': 0.056338028169014086, '(15, 20)':
↪  0.028169014084507043, '(75, 80)': 0.056338028169014086,
↪  '(60, 65)': 0.056338028169014086, '(70, 75)':
↪  0.056338028169014086, '(65, 70)': 0.056338028169014086,
↪  '(90, 95)': 0.0, '(80, 85)': 0.056338028169014086,
↪  '(85, 90)': 0.028169014084507043, '(30, 40)':
↪  0.1267605633802817, '(50, 60)': 0.1267605633802817,
↪  '(20, 30)': 0.1267605633802817, '(40, 50)':
↪  0.1267605633802817, '(10, 20)': 0.028169014084507043,
↪  '(70, 80)': 0.1267605633802817, '(60, 70)':
↪  0.1267605633802817, '(90, 100)': 0.0, '(80, 90)':
↪  0.09859154929577464, '(20, 40)': 0.2676056338028169,
↪  '(40, 60)': 0.2676056338028169, '(0, 20)':
↪  0.028169014084507043, '(60, 80)': 0.2676056338028169,
↪  '(80, 100)': 0.11267605633802817}, 'gender': {'*': 1,
↪  'Female': 0, 'Male': 0}, 'race': {'*': 1, 'Other': 0,
↪  'Amer-Indian-Eskimo': 0, 'Black': 0, 'White': 0,
↪  'Asian-Pac-Islander': 0}, 'marital_status': {'*': 1,
↪  'NM': 0.0, 'Married': 0.16666666666666666, 'leave':
↪  0.16666666666666666, 'alone': 0.16666666666666666,
↪  'Never-married': 0, 'Married-civ-spouse': 0,
↪  'Married-AF-spouse': 0, 'Divorced': 0, 'Separated': 0,
↪  'Widowed': 0, 'Married-spouse-absent': 0}}


metrica di loss: 2.0554451968758376

vettore delle generalizzazioni: (1, 0, 1, 2)

soppressione massima: 7

====================
Tabella:
             age  gender race marital_status
             ↪  occupation
```

```
0      (35, 40)    Male    *              *
↪  Adm-clerical
2      (35, 40)    Male    *              *
↪  Handlers-cleaners
10     (35, 40)    Male    *              *
↪  Exec-managerial
18     (35, 40)    Male    *              *
↪  Sales
22     (35, 40)    Male    *              *
↪  Farming-fishing
...          ...     ...  ...            ...
↪  ...
19045  (80, 85)  Female    *              *
↪  Other-service
19495  (80, 85)  Female    *              *
↪  Exec-managerial
19515  (80, 85)  Female    *              *
↪  Other-service
20482  (80, 85)  Female    *              *
↪  Adm-clerical
26731  (80, 85)  Female    *              *
↪  Prof-specialty

[30155 rows x 5 columns]
====================
```

Samarati su Bank Dataset

```
python main.py --samarati --k 3 --maxsup 10 --bank


configuration:
 {'k': 3, 'maxsup': 10, 'samarati': True, 'mondrian': False,
 ↪  'bank': True, 'data': {'path': 'data/bank.csv',
 ↪  'samarati_quasi_id': ['age', 'job'],
 ↪  'mondrian_quasi_id': ['age', 'job'], 'sensitive':
 ↪  'balance', 'columns': ['age', 'job', 'marital',
 ↪  'education', 'default', 'balance', 'housing', 'loan',
 ↪  'contact', 'day', 'month', 'duration', 'campaign',
 ↪  'pdays', 'previous', 'poutcome', 'y'],
 ↪  'samarati_generalization_type': {'age': 'range', 'job':
 ↪  'categorical'}, 'hierarchies': {'age': None, 'job':
 ↪  'data/bank_job.txt'}, 'mondrian_generalization_type':
 ↪  {'age': 'numerical', 'job': 'categorical'}}}

row count before sanitizing: 4520
row count sanitized: 4520

hierarchies:
```

```
 {'age': {'33': '(30, 35)', '35': '(35, 40)', '30': '(30,
 ↪  35)', '59': '(55, 60)', '36': '(35, 40)', '39': '(35,
 ↪  40)', '41': '(40, 45)', '43': '(40, 45)', '20': '(20,
 ↪  25)', '31': '(30, 35)', '40': '(40, 45)', '56': '(55,
 ↪  60)', '37': '(35, 40)', '25': '(25, 30)', '38': '(35,
 ↪  40)', '42': '(40, 45)', '44': '(40, 45)', '26': '(25,
 ↪  30)', '55': '(55, 60)', '67': '(65, 70)', '53': '(50,
 ↪  55)', '68': '(65, 70)', '32': '(30, 35)', '49': '(45,
 ↪  50)', '78': '(75, 80)', '23': '(20, 25)', '52': '(50,
 ↪  55)', '34': '(30, 35)', '61': '(60, 65)', '45': '(45,
 ↪  50)', '48': '(45, 50)', '57': '(55, 60)', '54': '(50,
 ↪  55)', '63': '(60, 65)', '51': '(50, 55)', '29': '(25,
 ↪  30)', '50': '(50, 55)', '27': '(25, 30)', '60': '(60,
 ↪  65)', '28': '(25, 30)', '21': '(20, 25)', '58': '(55,
 ↪  60)', '22': '(20, 25)', '46': '(45, 50)', '24': '(20,
 ↪  25)', '77': '(75, 80)', '75': '(75, 80)', '47': '(45,
 ↪  50)', '70': '(70, 75)', '65': '(65, 70)', '64': '(60,
 ↪  65)', '62': '(60, 65)', '66': '(65, 70)', '19': '(15,
 ↪  20)', '81': '(80, 85)', '83': '(80, 85)', '80': '(80,
 ↪  85)', '71': '(70, 75)', '72': '(70, 75)', '69': '(65,
 ↪  70)', '79': '(75, 80)', '73': '(70, 75)', '86': '(85,
 ↪  90)', '74': '(70, 75)', '76': '(75, 80)', '87': '(85,
 ↪  90)', '84': '(80, 85)', '(30, 35)': '(30, 40)', '(35,
 ↪  40)': '(30, 40)', '(55, 60)': '(50, 60)', '(40, 45)':
 ↪  '(40, 50)', '(20, 25)': '(20, 30)', '(25, 30)': '(20,
 ↪  30)', '(65, 70)': '(60, 70)', '(50, 55)': '(50, 60)',
 ↪  '(45, 50)': '(40, 50)', '(75, 80)': '(70, 80)', '(60,
 ↪  65)': '(60, 70)', '(70, 75)': '(70, 80)', '(15, 20)':
 ↪  '(10, 20)', '(80, 85)': '(80, 90)', '(85, 90)': '(80,
 ↪  90)', '(30, 40)': '(20, 40)', '(50, 60)': '(40, 60)',
 ↪  '(40, 50)': '(40, 60)', '(20, 30)': '(20, 40)', '(60,
 ↪  70)': '(60, 80)', '(70, 80)': '(60, 80)', '(10, 20)':
 ↪  '(0, 20)', '(80, 90)': '(80, 100)', '(20, 40)': '*',
 ↪  '(40, 60)': '*', '(60, 80)': '*', '(0, 20)': '*', '(80,
 ↪  100)': '*'}, 'job': {'white-collar': ' employed',
 ↪  'blue-collar': ' blue-collar', 'self-employed': '
 ↪  self-employed', 'admin.': ' white-collar', 'unknown': '
 ↪  other', 'unemployed': ' other', 'management': '
 ↪  white-collar', 'housemaid': ' blue-collar',
 ↪  'entrepreneur': ' self-employed', 'student': ' other',
 ↪  'retired': ' other', 'technician': ' blue-collar',
 ↪  'services': ' blue-collar'}}

hierarchy heights:
 {'age': 4, 'job': 0}

leaves_num:
```

```
 {'age': {'(30, 35)': 5, '(35, 40)': 5, '(55, 60)': 5, '(40,
  ↪  45)': 5, '(20, 25)': 5, '(25, 30)': 5, '(65, 70)': 5,
  ↪  '(50, 55)': 5, '(45, 50)': 5, '(75, 80)': 5, '(60,
  ↪  65)': 5, '(70, 75)': 5, '(15, 20)': 1, '(80, 85)': 4,
  ↪  '(85, 90)': 2, '(30, 40)': 10, '(50, 60)': 10, '(40,
  ↪  50)': 10, '(20, 30)': 10, '(60, 70)': 10, '(70, 80)':
  ↪  10, '(10, 20)': 1, '(80, 90)': 6, '(20, 40)': 20, '(40,
  ↪  60)': 20, '(60, 80)': 20, '(0, 20)': 1, '(80, 100)': 6,
  ↪  '*': 67}, 'job': {' employed': 2, 'employed': 1, '
  ↪  white-collar': 2, ' other': 4, ' blue-collar': 4, '
  ↪  self-employed': 2}}

loss_metric_map:
 {'age': {'*': 1, '33': 0, '35': 0, '30': 0, '59': 0, '36':
  ↪  0, '39': 0, '41': 0, '43': 0, '20': 0, '31': 0, '40':
  ↪  0, '56': 0, '37': 0, '25': 0, '38': 0, '42': 0, '44':
  ↪  0, '26': 0, '55': 0, '67': 0, '53': 0, '68': 0, '32':
  ↪  0, '49': 0, '78': 0, '23': 0, '52': 0, '34': 0, '61':
  ↪  0, '45': 0, '48': 0, '57': 0, '54': 0, '63': 0, '51':
  ↪  0, '29': 0, '50': 0, '27': 0, '60': 0, '28': 0, '21':
  ↪  0, '58': 0, '22': 0, '46': 0, '24': 0, '77': 0, '75':
  ↪  0, '47': 0, '70': 0, '65': 0, '64': 0, '62': 0, '66':
  ↪  0, '19': 0, '81': 0, '83': 0, '80': 0, '71': 0, '72':
  ↪  0, '69': 0, '79': 0, '73': 0, '86': 0, '74': 0, '76':
  ↪  0, '87': 0, '84': 0, '(30, 35)': 0.06060606060606061,
  ↪  '(35, 40)': 0.06060606060606061, '(55, 60)':
  ↪  0.06060606060606061, '(40, 45)': 0.06060606060606061,
  ↪  '(20, 25)': 0.06060606060606061, '(25, 30)':
  ↪  0.06060606060606061, '(65, 70)': 0.06060606060606061,
  ↪  '(50, 55)': 0.06060606060606061, '(45, 50)':
  ↪  0.06060606060606061, '(75, 80)': 0.06060606060606061,
  ↪  '(60, 65)': 0.06060606060606061, '(70, 75)':
  ↪  0.06060606060606061, '(15, 20)': 0.0, '(80, 85)':
  ↪  0.045454545454545456, '(85, 90)': 0.015151515151515152,
  ↪  '(30, 40)': 0.13636363636363635, '(50, 60)':
  ↪  0.13636363636363635, '(40, 50)': 0.13636363636363635,
  ↪  '(20, 30)': 0.13636363636363635, '(60, 70)':
  ↪  0.13636363636363635, '(70, 80)': 0.13636363636363635,
  ↪  '(10, 20)': 0.0, '(80, 90)': 0.07575757575757576, '(20,
  ↪  40)': 0.2878787878787879, '(40, 60)':
  ↪  0.2878787878787879, '(60, 80)': 0.2878787878787879,
  ↪  '(0, 20)': 0.0, '(80, 100)': 0.07575757575757576},
  ↪  'job': {'*': 1, 'white-collar': 0, 'blue-collar': 0,
  ↪  'self-employed': 0, 'admin.': 0, 'unknown': 0,
  ↪  'unemployed': 0, 'management': 0, 'housemaid': 0,
  ↪  'entrepreneur': 0, 'student': 0, 'retired': 0,
  ↪  'technician': 0, 'services': 0}}


metrica di loss: 0.28933360150172577

vettore delle generalizzazioni: (3, 0)
```

```
soppressione massima: 6

====================
Tabella:
            age       job balance
0     (20, 40)  services    4789
9     (20, 40)  services    9374
18    (20, 40)  services     132
38    (20, 40)  services     363
65    (20, 40)  services     338
...        ...       ...     ...
1680  (60, 80)   unknown     300
3876  (60, 80)   unknown     367
3962  (60, 80)   unknown    7337
4037  (60, 80)   unknown     353
4224  (60, 80)   unknown    4717

[4514 rows x 3 columns]
====================
```

Mondrian su Adult Dataset

```
python main.py --mondrian


configuration:
 {'k': 10, 'maxsup': 20, 'samarati': False, 'mondrian':
  ↪  True, 'bank': False, 'data': {'path':
  ↪  'data/adult.data', 'samarati_quasi_id': ['age',
  ↪  'gender', 'race', 'marital_status'],
  ↪  'mondrian_quasi_id': ['age', 'education_num'],
  ↪  'sensitive': 'occupation', 'columns': ['age',
  ↪  'work_class', 'final_weight', 'education',
  ↪  'education_num', 'marital_status', 'occupation',
  ↪  'relationship', 'race', 'gender', 'capital_gain',
  ↪  'capital_loss', 'hours_per_week', 'native_country',
  ↪  'class'], 'samarati_generalization_type': {'age':
  ↪  'range', 'gender': 'categorical', 'race':
  ↪  'categorical', 'marital_status': 'categorical'},
  ↪  'hierarchies': {'age': None, 'gender':
  ↪  'data/adult_gender.txt', 'race': 'data/adult_race.txt',
  ↪  'marital_status': 'data/adult_marital_status.txt'},
  ↪  'mondrian_generalization_type': {'age': 'numerical',
  ↪  'education_num': 'numerical'}}}

row count before sanitizing: 32561
row count sanitized: 30162

====================

metrica di loss: 0.2577096464479936
```

```
====================
Tabella:
          age education_num       occupation
209       17           3-7            Sales
262       17           3-7    Other-service
271       17           3-7    Other-service
335       17           3-7    Other-service
371       17           3-7    Other-service
...      ...           ...              ...
20483  74-90         15-16   Prof-specialty
21473  74-90         15-16   Prof-specialty
21835  74-90         15-16  Exec-managerial
23868  74-90         15-16  Exec-managerial
28176  74-90         15-16  Exec-managerial

[30162 rows x 3 columns]
====================
```

Mondrian su Bank Dataset

```
python main.py --mondrian --bank


configuration:
 {'k': 10, 'maxsup': 20, 'samarati': False, 'mondrian':
  ↪  True, 'bank': True, 'data': {'path': 'data/bank.csv',
  ↪  'samarati_quasi_id': ['age', 'job'],
  ↪  'mondrian_quasi_id': ['age', 'job'], 'sensitive':
  ↪  'balance', 'columns': ['age', 'job', 'marital',
  ↪  'education', 'default', 'balance', 'housing', 'loan',
  ↪  'contact', 'day', 'month', 'duration', 'campaign',
  ↪  'pdays', 'previous', 'poutcome', 'y'],
  ↪  'samarati_generalization_type': {'age': 'range', 'job':
  ↪  'categorical'}, 'hierarchies': {'age': None, 'job':
  ↪  'data/bank_job.txt'}, 'mondrian_generalization_type':
  ↪  {'age': 'numerical', 'job': 'categorical'}}}

row count before sanitizing: 4520
row count sanitized: 4520

====================

metrica di loss: 0.16187029719369647

====================
Tabella:
         age    job  balance
116   22-24    0-1     4111
270   22-24    0-1      174
444   22-24    0-1     1222
773   22-24    0-1      111
1003  22-24    0-1      204
...     ...    ...      ...
```

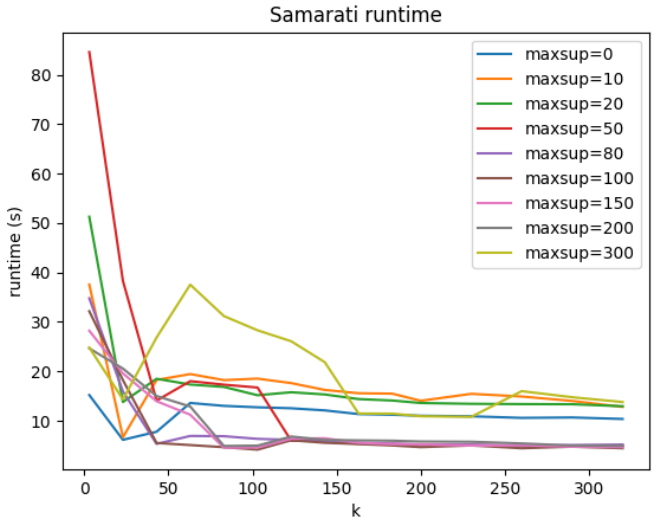
```
3876  58-71  10-11      367
3962  58-71  10-11     7337
4037  58-71  10-11      353
4136  58-71  10-11     3940
4224  58-71  10-11     4717

[4520 rows x 3 columns]
====================
```

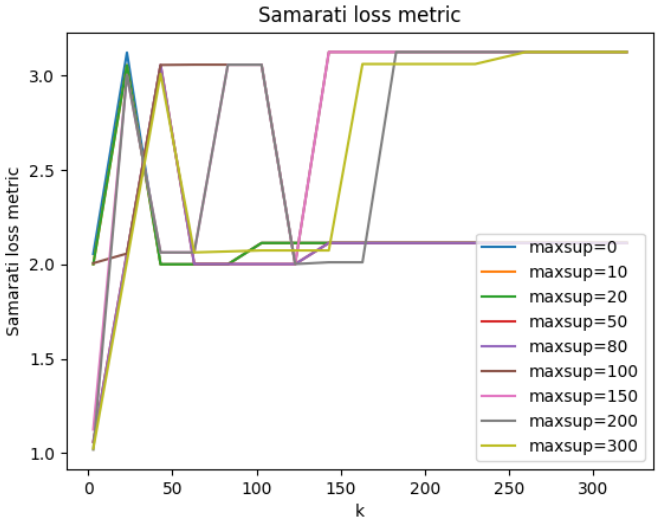
4.5.2 Plotting

Per visualizzare i risultati delle esecuzioni in termini di tempo di esecuzione e metrica di perdita in base ai differenti parametri scelti per l'esecuzione di entrambi gli algoritmi, sono stati generati dei grafici con la seguente configurazioni:

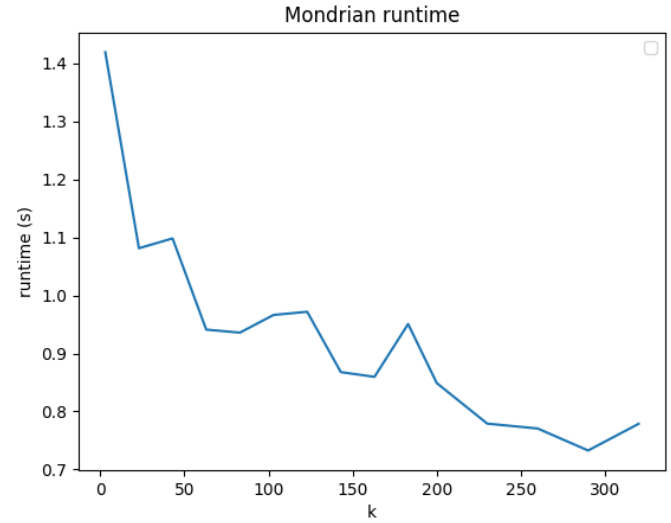
```
k_s_full = list(range(3, 200, 20)) + list(range(200, 350, 30))
maxsup_full = [0, 10, 20, 50, 80, 100, 150, 200, 300]
```

```
python plot.py --samarati
```

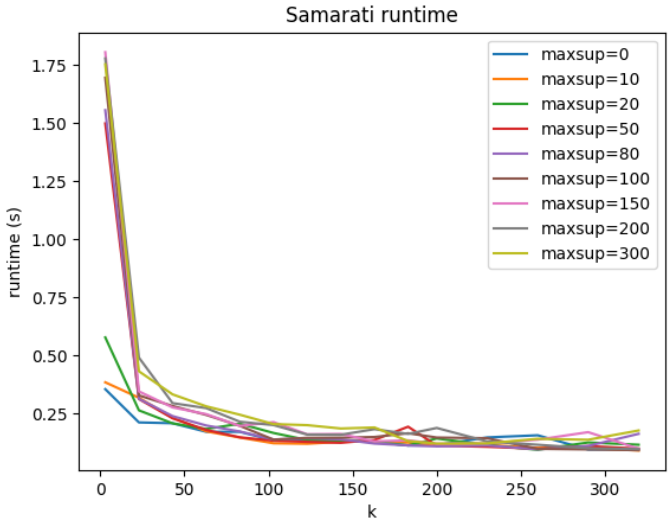


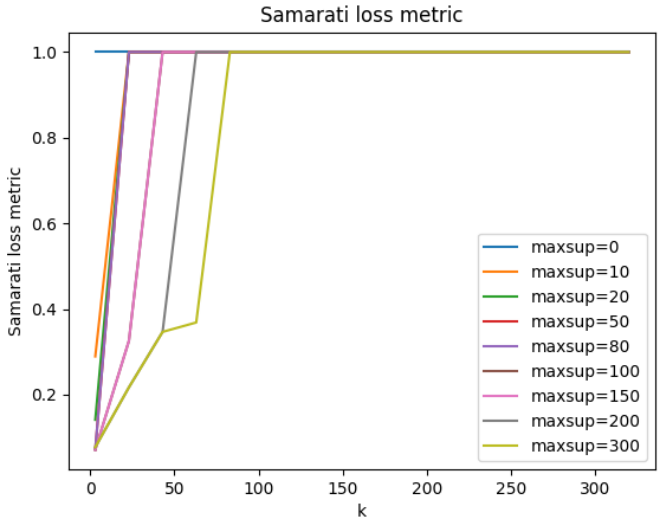


```
python plot.py --mondrian
```

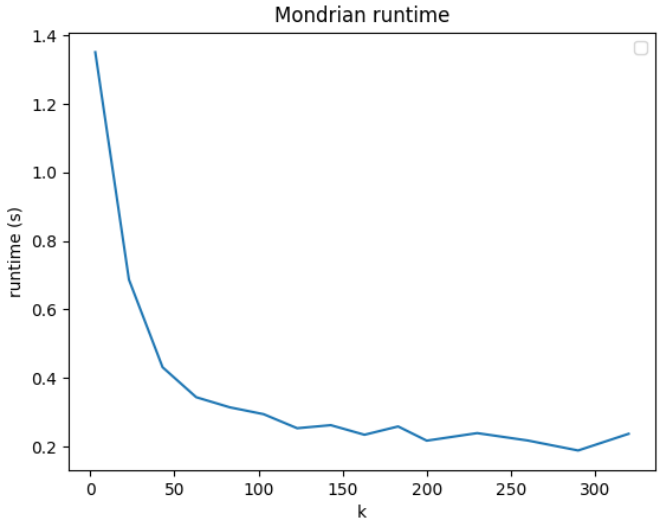


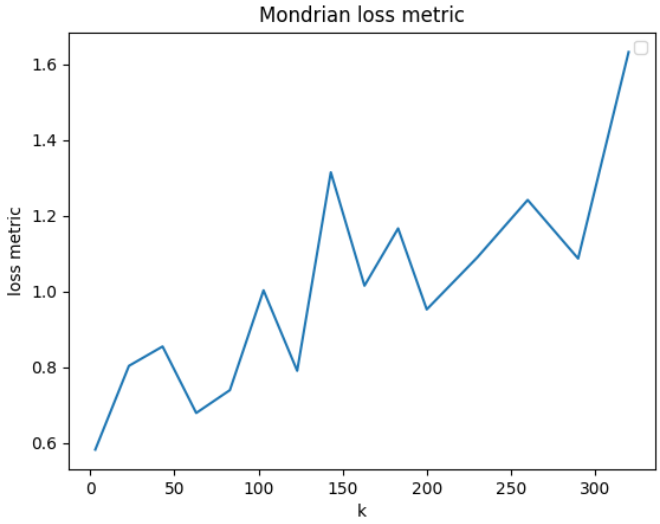
```
python plot.py --samarati --bank
```





```
python plot.py --mondrian --bank
```





4.6 Test di Performance di Accesso ai Dati

Per determinare l'impatto dell'anonimizzazione sulle prestazioni di accesso ai dati, è stato eseguito un test comparativo utilizzando un database SQLite salvato in memoria locale. Le query sono state eseguite attraverso l'attributo age con un valore di query specifico o eventualmente un intervallo equivalente per le tabelle anonimizzate, analizzandole nella tabella originale, nella tabella anonimizzata con l'algoritmo Samarati e nella tabella anonimizzata con l'algoritmo Mondrian.

4.6.1 Setup del primo Test

- **Database**: SQLite in memoria locale
- **Dataset**: *Adult*
- **Attributo di query d'esempio**: *age*
- **Valore di query d'esempio**: '39' (o intervallo equivalente)

Prime Righe dei Dati Originali:

```
   age        work_class  final_weight  education  education_num
   ↪  marital_status  ...  gender capital_gain capital_loss
   ↪  hours_per_week  native_country  class
0   39         State-gov         77516  Bachelors             13
↪  Never-married  ...    Male         2174            0             40
↪  United-States  <=50K
1   50  Self-emp-not-inc         83311  Bachelors             13
↪  Married-civ-spouse  ...    Male            0            0
↪  13   United-States  <=50K
2   38           Private        215646    HS-grad              9
↪  Divorced  ...    Male            0            0             40
↪  United-States  <=50K
3   53           Private        234721       11th              7
↪  Married-civ-spouse  ...    Male            0            0
↪  40   United-States  <=50K
4   28           Private        338409  Bachelors             13
↪  Married-civ-spouse  ...  Female            0            0
↪  40            Cuba  <=50K
```

Anonimizzati (Samarati):

```
       age gender race marital_status         occupation   work_class
       ↪  ...   relationship capital_gain  capital_loss hours_per_week
       ↪  native_country  class
0  (35, 40)   Male    *              *       Adm-clerical    State-gov
↪  ...  Not-in-family         2174             0             40
↪  United-States  <=50K
1  (35, 40)   Male    *              *  Handlers-cleaners      Private
↪  ...  Not-in-family            0             0             40
↪  United-States  <=50K
2  (35, 40)   Male    *              *    Exec-managerial      Private
↪  ...        Husband            0             0             80
↪  United-States   >50K
```

```
3  (35, 40)   Male    *              *              Sales      Private
↪  ...        Husband           0            0            50
↪  United-States  <=50K
4  (35, 40)   Male    *              *    Farming-fishing  Federal-gov
↪  ...        Husband           0            0            40
↪  United-States  <=50K
```

Anonimizzati (Mondrian):

```
    age education_num  occupation work_class   final_weight education
    ↪  marital_status  ... race    gender  capital_gain
    ↪  capital_loss  hours_per_week  native_country  class
0  17-19           1-7      Other-service    Private        309634
↪  11th  Never-married  ...  White  Female           0            0
↪  22   United-States  <=50K
1  17-19           1-7  Handlers-cleaners    Private         25828
↪  11th  Never-married  ...  White    Male           0            0
↪  16   United-States  <=50K
2  17-19           1-7              Sales    Private        140164
↪  11th  Never-married  ...  White  Female           0            0
↪  40   United-States  <=50K
3  17-19           1-7              Sales    Private         65368
↪  11th  Never-married  ...  White  Female           0            0
↪  12   United-States  <=50K
4  17-19           1-7      Other-service    Private        245918
↪  11th  Never-married  ...  White    Male           0            0
↪  12   United-States  <=50K
```

Query Eseguite

1. **Tabella originale**:

```
SELECT * FROM original WHERE age = '39'
```

2. **Tabella anonimizzata (Samarati)**:

```
SELECT * FROM anonymizedS WHERE "age" = "(35, 40)"
```

3. **Tabella anonimizzata (Mondrian)**:

```
SELECT * FROM anonymizedM WHERE "age" = "39" OR "age" = "38-47" OR "age" = "38-48"
```

Risultati del Test Tempi medi di accesso ai dati:

- Tabella originale: 0.0068 secondi
- Tabella anonimizzata con Samarati: 0.0167 secondi
- Tabella anonimizzata con Mondrian: 0.0066 secondi

Numero di tuple restituite:

- Tabella originale: 786
- Tabella anonimizzata con Samarati: 4085
- Tabella anonimizzata con Mondrian: 956

4.6.2 Setup del secondo Test

- **Database**: SQLite in memoria locale
- **Dataset**: *Bank*
- **Attributo di query d'esempio**: *age*
- **Valore di query d'esempio**: '33' (o intervallo equivalente)

Prime Righe dei Dati Originali:

```
      age          job  marital  education default  balance housing loan
       ↪  contact  day month  duration  campaign  pdays  previous
       ↪  poutcome   y
0   33     services  married  secondary      no     4789     yes  yes
↪  cellular   11   may       220         1    339         4  failure
↪  no
1   35   management   single   tertiary      no     1350     yes   no
↪  cellular   16   apr       185         1    330         1  failure
↪  no
2   30   management  married   tertiary      no     1476     yes  yes
↪  unknown    3   jun       199         4     -1         0  unknown  no
3   59  blue-collar  married  secondary      no        0     yes   no
↪  unknown    5   may       226         1     -1         0  unknown  no
4   35   management   single   tertiary      no      747      no   no
↪  cellular   23   feb       141         2    176         3  failure
↪  no
```

Anonimizzati (Samarati):

```
        age          job  balance  marital  education default housing
         ↪  loan   contact  day month  duration  campaign  pdays
         ↪  previous poutcome   y
0   *     services     4789  married  secondary      no     yes  yes
↪  cellular   11   may       220         1    339         4  failure
↪  no
1   *   management     1350   single   tertiary      no     yes   no
↪  cellular   16   apr       185         1    330         1  failure
↪  no
2   *   management     1476  married   tertiary      no     yes  yes
↪  unknown    3   jun       199         4     -1         0  unknown  no
3   *  blue-collar        0  married  secondary      no     yes   no
↪  unknown    5   may       226         1     -1         0  unknown  no
4   *   management      747   single   tertiary      no      no   no
↪  cellular   23   feb       141         2    176         3  failure
↪  no
```

Anonimizzati (Mondrian):

```
     age   job marital  balance  education default housing loan
      ↪  contact  day month  duration  campaign  pdays  previous
      ↪  poutcome   y
0  23-32  0-11     0-1     1476   tertiary      no     yes  yes
↪  unknown    3   jun       199         4     -1         0  unknown  no
```

```
1  23-32  0-11     0-1      360  secondary      no     yes  yes
↪  cellular   29   jan        89         1    241         1  failure
↪  no
2  23-32  0-11     0-1      132  secondary      no      no   no
↪  cellular    7   jul       148         1    152         1    other
↪  no
3  23-32  0-11     0-1      543   tertiary      no      no   no
↪  cellular   30   jan       169         3     -1         0  unknown
↪  no
4  23-32  0-11     0-1      171  secondary      no      no   no
↪  cellular   27   aug        81         3     -1         0  unknown
↪  no
```

Query Eseguite

1. **Tabella originale**:

   ```
   SELECT * FROM original WHERE age = '33'
   ```

2. **Tabella anonimizzata (Samarati)**:

   ```
   SELECT * FROM anonymizedS WHERE "age" = "*"
   ```

3. **Tabella anonimizzata (Mondrian)**:

   ```
   SELECT * FROM anonymizedM WHERE "age" = "33-35"
   ```

Risultati del Test Tempi medi di accesso ai dati:

- Tabella originale: 0.0009 secondi
- Tabella anonimizzata con Samarati: 0.0163 secondi
- Tabella anonimizzata con Mondrian: 0.0015 secondi

Numero di tuple restituite:

- Tabella originale: 186
- Tabella anonimizzata con Samarati: 4520
- Tabella anonimizzata con Mondrian: 415

4.6.3 Analisi Comparativa

1. **Efficienza di accesso**:
   - L'anonimizzazione con l'algoritmo Samarati ha riportato un incremento del tempo di accesso di circa 2 volte e mezzo più alto rispetto alla tabella originale.
   - L'algoritmo Mondrian invece ha mantenuto tempi di accesso paragonabili a quelli della tabella non anonimizzata.

- La differenza nei tempi di accesso può essere dovuta alla struttura di generalizzazione: la generalizzazione basata su cluster di Mondrian consente una partizione più efficiente dei dati, riducendo così il numero di tuple restituite.

2. **Precisione dei risultati**:

   - Entrambi gli algoritmi di anonimizzazione hanno visto un incrementato del numero di tuple restituite a causa del processo di generalizzazione.
   - L'algoritmo Samarati ha restituito circa 5 volte in più del numero di tuple rispetto alla tabella originale.
   - L'algoritmo Mondrian ha prodotto un incremento più contenuto, mantenendo un numero di tuple restituite più vicino a quello della tabella originale, in quanto Mondrian attraverso i cluster consente una partizione più efficiente dei dati, riducendo così l'aumento di tuple analizzate.

3. **Qualità dei dati**:

   - La qualità dei dati anonimizzati con l'algoritmo Mondrian è risultata quindi superiore, poiché la generalizzazione per cluster preserva maggiormente le informazioni specifiche rispetto alla generalizzazione basata su intervalli dell'algoritmo Samarati.
   - L'algoritmo Samarati, pur garantendo un buon livello di anonimato, tende a generare intervalli più ampi che aumentano il numero di tuple restituite nelle query, riducendo la precisione delle risposte.

4. **Potenziale scalabilità**:

   - Generalmente l'algoritmo Mondrian ha dimostrato un'efficienza maggiore, soprattutto per dataset di grandi dimensioni, grazie alla sua capacità di ridurre il numero di tuple restituite.
   - L'algoritmo Samarati potrebbe essere invece più indicato per dataset di dimensioni ridotte dove la perdita di precisione è meno critica, anche se l'eccessiva generalizzazione potrebbe essere in certi casi controproducente (come nel caso del dataset Bank)

Il test di performance di accesso ai dati ha evidenziato come l'algoritmo di anonimizzazione scelto possa influire in modo significativo sulle prestazioni e sulla qualità dei risultati. In particolare:

- L'algoritmo Mondrian ha dimostrato un miglior bilanciamento tra tempi di accesso, numero di tuple restituite e qualità delle informazioni anonimizzate tramite k-anonimato, risultando più adatto per scenari con grandi dataset e dove vi è quindi necessità di query efficienti.

- L'algoritmo Samarati, pur garantendo un ottimo livello di K-anonimity, comporta in certi casi un aumento significativo dei tempi di accesso e una maggiore dispersione delle risposte alle query di accesso a causa della sua generalizzazione più ampia.

In generale, l'algoritmo utilizzato per la gestione del k-anonimato deve tenere conto delle specifiche esigenze e scenari applicativi, bilanciando eventualmente la necessità di anonimato con l'efficienza di accesso e la precisione dei dati restituiti dalle query.

4.7 Conclusione e Analisi dei risultati

L'esecuzione degli algoritmi di k-anonimato, Samarati e Mondrian, sui dataset Adult e Bank dimostra come entrambi i metodi funzionino efficacemente nell'anonimizzazione dei dati rispettando tutti i vincoli del k-anonimato.

L'analisi delle prestazioni basata su vari valori di k e maxsup evidenzia che la scelta di questi parametri influenza significativamente il trade-off tra anonimato e perdita di dati. Nello specifico, valori più alti di k tendono a migliorare l'anonimato ma possono aumentare la perdita di dati, riducendo così la granularità e l'utilità dei dati anonimi. Al contrario, valori più bassi di k mantengono una maggiore utilità dei dati ma, se non scelti correttamente, possono compromettere l'anonimato, aumentando il rischio di re-identificazione.

Gli algoritmi hanno fornito risultati variabili a seconda delle caratteristiche specifiche dei dataset e dei parametri scelti, indicando l'importanza di una configurazione attenta e mirata a seconda del contesto e dello scenario applicativo. Ad esempio, nei dataset con attributi altamente distintivi, è necessaria una configurazione più prudente per evitare la perdita di anonimato (come nel dataset adult), mentre in dataset più piccoli o in cui è già presente anonimizzazione è possibile utilizzare una configurazione meno aggressiva. Inoltre, la variabilità nei risultati suggerisce che, come ovvio, non esiste una soluzione unica che sia ottimale per tutti i tipi di dataset, sottolineando la necessità di un approccio personalizzato in base agli attributi e i record presi in campione.

In conclusione, entrambi gli algoritmi rappresentano validi strumenti per ottenere il k-anonimato. Tuttavia, è importante evidenziare che l'algoritmo Mondrian si distingue per la sua maggiore velocità di esecuzione rispetto all'algoritmo Samarati. Questa caratteristica rende Mondrian particolarmente adatto per applicazioni che richiedono tempi di risposta rapidi. Inoltre, l'algoritmo Mondrian ha dimostrato di raggiungere una metrica di perdita inferiore, mantenendo così una maggiore utilità dei dati anonimizzati. Pertanto, sebbene entrambi gli algoritmi siano efficaci, Mondrian potrebbe essere preferibile in diversi scenari.

5 Scenari

5.1 Scenari Applicativi

Non sempre è possibile applicare il k-anonimato e i suoi rafforzamenti, come l-diversità e t-chiusura, in ogni contesto. Ad esempio non è possibile ottenere il K-anonimato:

- In presenza di molteplici tuple che fanno riferimento a singoli soggetti.
- Nei casi di rilasci multipli di tabelle interdipendenti (tabelle che sono in continuo aggiornamento).
- Quando sono presenti numerosi quasi-identificatori.
- Quando non è possibile definire precisamente i quasi-identificatori di una tabella.
- Nelle situazioni di rilascio di dati in stream anziché di dataset completi (rilascio dati non appena vengono acquisiti).
- In presenza di eventuali restrizioni sulla privacy degli utenti.

Tuttavia, esistono scenari in cui viene standardizzato un approccio relativo al k-anonimato.

5.1.1 Social Network

I social network si basano solitamente su database di tipo grafico, dove il grafo sociale rappresenta le relazioni di follow degli utenti, con i nodi che corrispondono agli utenti stessi e gli archi che rappresentano le relazioni tra di essi. L'anonimato in un grafo sociale non è sempre garantito, anche se vengono applicate tecniche per mascherare i nominativi e i dati personali degli utenti in alcuni è ancora possibile rilevare l'identità degli utenti attraverso tecniche di ingegneria sociale (ad esempio, analizzando il conteggio degli archi uscenti dal nodo, che può corrispondere al numero di relazioni dell'utente). Per garantire il k-anonimato in questo contesto, è essenziale rendere indistinguibili i nodi del grafo. Ciò implica che almeno k nodi devono avere lo stesso numero di archi che li collegano (eventualmente aggiungendo archi fittizi, chiamati "fake edges", per raggiungere il valore k).

Anonimato nei grafi 1 *Un vertice u è k-anonimo se esistono almeno $k-1$ vertici $v_1, v_2, \ldots, v_{k-1}$ tali che i sotto-grafi creati dai nodi vicini a u e ai suoi vicini $v_1, v_2, \ldots, v_{k-1}$ siano isomorfi. Un grafo rispetta il k-anonimato se tutti i suoi vertici sono k-anonimi.*

5.1.2 Data Mining / Big Data Analytics

Il Data Mining è un processo di estrazione di informazioni di larga utilità da insiemi di dati estremamente vasti. Tipicamente, si avvale di tecniche quali l'analisi statistica, algoritmi di machine learning e tecniche di visualizzazione dei dati. Il Data Mining identifica pattern, trend e relazioni nascoste nei dati. Nel contesto del Data Mining, esistono due approcci principali per rendere anonimi i dati:

1. *Anonymize and Mine*: Questo approccio, generalmente conservativo, è adottato quando l'analisi dei dati non è eseguita da un soggetto fidato. Pertanto, è necessario anonimizzare preventivamente i dati al fine di preservare la privacy degli utenti. Tuttavia, questo metodo tende ad essere poco utilizzato a causa del suo elevato costo computazionale.

2. *Mine and Anonymize*: Questo approccio, di solito preferito, prevede che l'analisi dei dati sia eseguita da soggetto di fiducia, come un'organizzazione esterna fidata o un internamente. In questo caso, i dati sono anonimizzati dopo l'analisi e prima della loro pubblicazione. Tale metodologia è preferibile in quanto consente di eseguire l'analisi sui dati originali e non generalizzati, inoltre comporta un minor costo computazionale.

5.1.3 Dati di localizzazione

I dati di localizzazione (informazioni spaziali degli individui) consentono di tracciare la posizione degli utenti presenti nella base di dati. In questo contesto, il K-anonimato viene impiegato per proteggere l'identità delle persone rispetto alla loro posizione associata. Per attuare la k-anonimità, è necessario generalizzare le aree a cui sono associati gli utenti, in modo da ottenere un'area di riferimento che includa almeno $k-1$ ulteriori utenti. La privacy dei dati relativi alla localizzazione si suddivide in due macro-categorie:

- *Privacy di localizzazione*: Questa categoria riguarda specificamente la posizione precisa associata a ciascun utente nella base di dati ovvero riguarda in particolare i LBs - Location Based Services. Per garantire la privacy degli utenti, è fondamentale evitare di indicare una posizione precisa e univoca per ciascun individuo, in particolare se la posizione costituisce un quasi-identificatore. In questo caso, è essenziale generalizzare la posizione in uno spettro più ampio, in modo da includere un numero sufficiente di utenti per soddisfare il parametro k e rispettare così il K-anonimato. Questa tipologia di privacy di locazione è poi suddivisibile in altri sotto-scenari nei quali è possibile applicare diverse tecniche in base al tipo di protezione che si vuole ottenere (ad esempio: veicoli a guida autonoma, applicazioni di navigazione stradale, ...)

- *Privacy della traiettoria*: Questa categoria riguarda i dati relativi ai percorsi seguiti dagli utenti. Nel caso in cui tali percorsi possano essere utilizzati per identificare gli utenti nella base di dati, diventando così un quasi-identificatore, diventa necessario garantire l'anonimato. Ciò può essere ottenuto mediante tecniche di blocco del tracciamento, che consentono di modificare e mescolare i percorsi seguiti dagli utenti in modo che il percorso o parte del percorso sia condiviso con almeno altri $k-1$ utenti al fine di preservare la loro privatezza.

6 Privacy Differenziale

Non sempre il k-anonimato rappresenta la soluzione ottimale per l'anonimizzazione di una base di dati, né è sempre possibile applicare le tecniche necessarie per garantire il rispetto delle metriche del k-anonimato in un particolare database. Questa tecnica può quindi essere definita come un punto di partenza per la protezione della privacy degli utenti, ma non è l'unica applicabile nel contesto della protezione dei dati. Infatti, esiste anche la privatezza differenziale che potrebbe essere applicata singolarmente oppure congiuntamente al K-anonimato.

6.1 Privatezza differenziale

La privatezza differenziale utilizza un approccio differente rispetto al k-anonimato, perturbando il risultato dell'analisi su dataset non protetti [3]. In particolare, la privatezza differenziale mira a prevenire che eventuali osservatori possano rilevare la presenza o meno di un individuo specifico in un insieme di dati. Questa tecnica implementa meccanismi, applicabili durante il rilascio dei dati, richiedendo che la distribuzione di probabilità dei dati pubblicati sia la stessa, indipendentemente dal fatto che un individuo sia presente o meno nel set di dati. In altre parole, garantisce che per ogni coppia di dataset che differiscono al massimo per una tupla, la probabilità di osservare un certo risultato nel primo dataset sia molto vicina alla probabilità di osservare lo stesso risultato nel secondo dataset [2]. Solitamente, per applicare questa tecnica, si aggiunge del rumore casuale nei dati tramite una funzione randomizzata, perturbando di conseguenza la veridicità dei dati. La privatezza differenziale è utile quando è necessaria la protezione della privacy in DBMS statistici (esecuzione di query in tempo reale) oppure su dati statistici (rilascio di basi di dati pre-computate).

7 Conclusioni

7.1 K-Anonimato e Algoritmi di K-Anonimato

Il k-anonimato rappresenta quindi un approccio fondamentale nella protezione dei dati sensibili, garantendo che ogni record in un dataset non possa essere distinto da almeno $k-1$ altri record sulla base degli attributi quasi-identificatori. Questa tecnica, nonostante le sue limitazioni, come la vulnerabilità a vari attacchi se non implementata in modo corretto e con scarsa attenzione, è essenziale per la protezione della privacy in numerosi contesti.

7.2 Algoritmi di K-Anonimato

Nella ricerca sono stati esaminati vari algoritmi per implementare il k-anonimato:

7.2.1 Algoritmi di Generalizzazione

- Basata sulla gerarchia: sostituisce i valori degli attributi QI con versioni più generiche seguendo una gerarchia predefinita.
- Basata sui record: generalizza i dati in base alla similarità tra i record, cercando di mantenere il più possibile l'informazione.

7.2.2 Algoritmi di Frammentazione e Microaggregazione:

- Frammentazione dei dati: divide i dati in blocchi più piccoli e li divide in sottotabelle per ridurre il rischio di re-identificazione.
- Microaggregazione: raggruppa i record in piccoli cluster e sostituisce i valori dei QI calcolando le varie medie dei cluster.

7.3 Test degli algoritmi

Il progetto[7] ha esaminato in modo approfondito il concetto di k-anonimato e due dei principali algoritmi utilizzati per ottenerlo, in particolare: Samarati e Mondrian. Questi algoritmi sono stati implementati e testati utilizzando due dataset distinti: l'Adult Dataset e il Bank Dataset, al fine di valutare l'efficacia e le prestazioni di ciascuna metodologia.

7.3.1 Algoritmo di Samarati

Algoritmo che si basa su un approccio di generalizzazione gerarchica, valutando le varie possibili soluzioni e selezionando quella che minimizza la perdita di informazioni $maxsup$ pur mantenendo il k-anonimato k. Utilizza una struttura ad albero per esplorare le combinazioni di generalizzazione presenti nella gerarchia.

7.3.2 Algoritmo di Mondrian

Implementa una suddivisione spaziale multi-dimensionale, trattando i quasi-identificatori come dimensioni di uno spazio e suddividendolo in regioni che soddisfano il criterio di k-anonimato. È risultato nei test migliore in velocità di esecuzione e perdita di informazioni rispetto all'algoritmo di Samarati.

7.3.3 Risultati dell'Esecuzione

In generali i test condotti hanno utilizzato specifiche configurazioni di parametri per ciascun algoritmo e dataset, con l'obiettivo di ottimizzare l'efficacia dell'anonimizzazione e minimizzare la perdita di informazioni. I risultati ottenuti dagli esperimenti ha rivelato che valori più alti di k migliorano l'anonimato ma aumentano la perdita di dati, riducendo quindi l'utilità dei dati anonimizzati. Valori più bassi di k permettono di mantenere una maggiore utilità dei dati ma possono compromettere l'anonimato se non vengono scelti correttamente i QI, il parametro k. Per quanto riguarda le prestazioni, l'algoritmo di Mondrian ha dimostrato maggiore velocità nei tempi di esecuzione e minore perdita di informazioni rispetto all'algoritmo Samarati.

8 Bibliografia

References

[1] V. Ciriani et al. "κ-Anonymity". In: *Secure Data Management in Decentralized Systems*. Ed. by Ting Yu and Sushil Jajodia. Boston, MA: Springer US, 2007, pp. 323–353. ISBN: 978-0-387-27696-0. DOI: 10.1007/978-0-387-27696-0_10. URL: https://doi.org/10.1007/978-0-387-27696-0_10.

[2] S. De Capitani di Vimercati et al. "k-Anonymity: From Theory to Applications". In: *Transactions on Data Privacy* 16.1 (Jan. 2023), pp. 25–49.

[3] *Differential privacy*. Vol. 2006. ICALP, 2006, pp. 1–12. URL: https://link.springer.com/chapter/10.1007/11787006_1.

[4] Wei Jiang and Chris Clifton. "Privacy-Preserving Distributed k-Anonymity". In: *Data and Applications Security XIX*. Ed. by Sushil Jajodia and Duminda Wijesekera. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 166–177. ISBN: 978-3-540-31937-5.

[5] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *2007 IEEE 23rd international conference on data engineering*. IEEE. 2006, pp. 106–115.

[6] ltzheng Longtao Zheng. *data-privacy*. 2021. URL: https://github.com/ltzheng/data-privacy.git.

[7] Molinari Lorenzo. *Anonimato e K-anonimity*. 2024. URL: https://github.com/LoreMolinari/Anonimato-K-anonymity.git.

[8] Ashwin Machanavajjhala et al. "L-diversity: Privacy beyond k-anonymity." In: *ACM Trans. Knowl. Discov. Data* 1.1 (2007), p. 3.

[9] P. Samarati. "Protecting Respondents' Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 13.6 (Nov. 2001), pp. 1010–1027.

[10] UCI Machine Learning Repository. *Adult Data Set*. 1994. URL: https://archive.ics.uci.edu/ml/datasets/adult.

[11] UCI Machine Learning Repository. *Bank Marketing Data Set*. 2014. URL: https://archive.ics.uci.edu/dataset/222/bank+marketing.

[12] Ke Wang, Benjamin C. M. Fung, and Guozhu Dong. "Integrating Private Databases for Data Analysis". In: *Intelligence and Security Informatics*. Ed. by Paul Kantor et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 171–182. ISBN: 978-3-540-32063-0.

[13] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. "Privacy-enhancing k-anonymization of customer data". In: *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '05. ¡conf-loc¿, ¡city¿Baltimore¡/city¿, ¡state¿Maryland¡/state¿, ¡/conf-loc¿: Association for Computing Machinery, 2005, pp. 139–147. ISBN: 1595930620. DOI: 10.1145/1065167.1065185. URL: https://doi.org/10.1145/1065167.1065185.