Risposte orale - Federico Torrielli

Privacy by default e privacy by design

Privacy by default significa che in ogni sistema dove deve essere garantita la privacy, essa deve essere messa sempre "al massimo". Ovvero, quando proposta una scelta, il valore di default deve essere quello privacy-respecting.

Privacy by design è un concetto introdotto nel 2012 da Ann Cavoukian e consiste nell'integrale in maniera fondamentale, già dal principio di ogni sistema o applicazione, il concetto della privacy. Esso comprende 7 principi fondamentali tra cui:

- Proattività e non reattività (prevenire è meglio che curare)
- Privacy come impostazione di default
- Privacy nel design del sistema
- Somma positiva, e non zero (sistemi win-win)
- Visibilità e trasparenza
- Creazione di sistemi user-centric

Novità del GDPR

Il GDPR integra diverse novità rispetto alla direttiva europea per la privacy.

- Pseudonimizzazione
- Responsabilità e responsabilizzazione

Pseudonimizzazione come column encryption oppure dynamic data masking

Responsabilità come nuovi ruoli (titolare del trattamento dei dati, responsabile del trattamento eDPO), trasparenza ed auditing

Responsability vs accountability

- Responsability (responsabilità): il compito di rispondere e completare controlli
- Accountability (responsabilizzazione): l'abilità o il dovere di riportare eventi, controlli e problemi –> Si applica dopo che un evento è capitato

Data breach, cosa fare

In caso di data breach, a seconda della gravità del fatto, il titolare del trattamento dei dati deve informare le autorità entro 72 ore. Se l'impatto è negativo sugli utenti, essi dovranno essere informati.

Nel caso contrario, ci sono sanzioni: avvertimento scritto, audit periodici, 10 mln di euro o 2% del fatturato oppure 20 mln di euro o 4% del fatturato in caso di gravi violazioni

Regole di trasferimento dati UE-USA

- Safe harbor privacy principles (2000 - 2015)
- USA-EU privacy shield

RBAC vs ABAC

- RBAC: autorizzazione basata sui ruoli (ok per piccole aziende)
- ABAC: autorizzazione basata sugli attributi (migliore, controllo a grana fine)

Auditing

Processo di validazione ed esame di documenti, processi, dati, procedure e sistemi.

Il processo produce un log di audit, ed esso può essere fatto in modo interno (la stessa azienda) oppure in modo esterno (un'azienda dedicata all'auditing)

Processo: Planning –> Definizione obiettivi –> Review, verifica e validazione –> Report e documentazione

K-Anonimity

Framework che attraverso la generalizzazione e la soppressione ha il fine di nascondere un'individuo in una tabella tra altri k-1 individui, e previene da il linking attack con una confidenza $> 1/k$.

Concetti fondamentali: quasi-identifier, gerarchia di generalizzazione del dominio e del valore

Limiti della K-Anonimity:

- Homogeneity attack: Potrebbe accadere che tutte le tuple con lo stesso valore QI in una tabella hanno lo stesso valore nell'attributo sensibile.
- Background knowledge attack che può portare ad una positive o negative disclosure

Scelta del K

Il problema della K-Anonimity è scegliere un k adatto alla singola situazione per evitare di generalizzare troppo: la perdita di informazioni è proporzionale alla grandezza del k. Non esiste un k per ogni problema, ma i seguenti ambienti sono caratterizzanti della scelta:

- Grandezza del dataset
- Variabilità del dominio
- Numero di attributi per record
- Presenza di QI molto simili o meno

Problemi principali della K-anonimity

- Non è applicabile a dataset enormi (curse of dimensionality)
- Spesso è complessa (vd. Incognito) e computazionalmente problematica
- Introduce del lavoro aggiuntivo che deve essere fatto dalla persona, ovvero distinguere QI da attributi sensibili (non è banale, se sono molto vicini!)
- Non conta la mancanza di diversità su una tabella (vd. l-diversity)

Algoritmi della k-anonimity

Algoritmo di Samarati

Considerazioni iniziali:

- Ogni percorso nel reticolo DGH_{DT} rappresenta una strategia di generalizzazione che va dalla parte bottom alla parte top del reticolo stesso;
- Ci si basa sulla **generalizzazione locale minima** ovvero il nodo più basso di ogni percorso che soddisfa la k-anonimity per ogni percorso;
- Altre proprietà:
  - Ogni generalizzazione K-minima è localmente minima rispetto al percorso, tuttavia non è vero il contrario.
  - Man mano che si sale nella gerarchia il numero di tuple che devono essere rimosse per garantire la k-anonymity diminuisce: conseguenza del fatto che sono state rimosse le cosiddette tuple outliner che potrebbero compromettere la k-anonimity del dataset.
- Se non c'è una soluzione che garantisce la k-anonimity sopprimendo meno tuple di MaxSup all'altezza h, non può esistere una soluzione con altezza < di h che la garantisca.

L'ultima proprietà viene sfruttata con un approccio dicotomico sul reticolo dei distance vectors. L'algoritmo quindi sarà:

1. Valutare soluzioni all'altezza h/2;
2. Se esiste almeno una soluzione che soddisfa la k-anonimity:
   a. Allora valuta le soluzioni ad altezza ⌊ h/4 ⌋;
   b. Altrimenti valuta le soluzioni ad altezza ⌊ 3h/4 ⌋.
3. Proseguire fin quando l'algoritmo non raggiunge l'altezza minima per cui c'è un distance vector che soddisfa la k-anonimity.

Per ridurre il costo computazionale, L'algoritmo adotta una matrice dei vettori di distanza la quale evita il calcolo esplicito di ogni tabella generalizzata.

Algoritmo Incognito

Idea: la k-anonimity, rispetto ad un sottoinsieme proprio di QI, è una condizione necessaria, ma non sufficiente per la k-anonimity, rispetto a QI. Con questo vogliamo dire che se prendiamo un sottoinsieme dell'insieme dei QI e al suo interno non è rispettata la k-anonimity allora non sarà rispettata anche nell' insieme generale.

Funzionamento:

- **Iterazione 1**: controlla la k-anonimity per ciascun attributo in QI, scartando la generalizzazione che non la soddisfa;
- **Iterazione 2**: combina le restanti generalizzazioni in coppie e controlla la k-anonimity per ogni coppia ottenuta;
- **Iterazione i**: combina tutte le i-uple di attributi ottenute combinando generalizzazioni che soddisfacevano la k-anonimity all'iterazione i –1. Elimina le soluzioni non anonime;
- **Iterazione | QI |**: restituisce il risultato finale.

Approccio Bottom-Up: parte da elementi meno generalizzanti fino ad elementi più generalizzanti per la visita del DHGs. Inoltre, questo algoritmo adotta un approccio a priori, cioè basato sulla generalizzazione dei candidati.

Inoltre, sfrutta le proprietà di monotonicità relative alla frequenza delle tuple nel reticolo (ricorda le gerarchie OLAP e il frequente mining del set di elementi).

Algoritmo Mondrian

Considerazioni iniziali:

- Ogni attributo in QI rappresenta una dimensione;
- Ogni tupla in PT (tabella privata) è rappresentata da un punto nello spazio definito da QI;
- Le tuple con lo stesso valore QI sono rappresentate associando il numero di occorrenze con dei punti;
- Lo spazio multidimensionale è partizionato dividendo le dimensioni in modo tale che ogni area contenga almeno k occorrenze dei valori dei punti;
- Tutti i punti in una regione sono generalizzati ad un valore unico;
- Le tuple corrispondenti sono sostituite da generalizzazioni calcolate.

Per raggruppare occorre utilizzare un criterio.

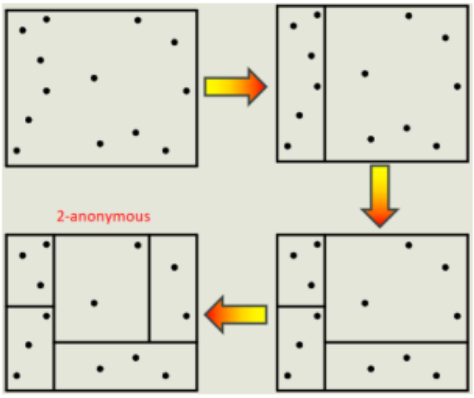
A differenza degli algoritmi visti in precedenza samarati e incognito (lavorano per minimizzare il livello di generalizzazione), tale algoritmo si usa la **discernability metric**.

Essa penalizza ogni tupla con le dimensioni del gruppo a cui appartiene (per grandezza intendiamo quanti valori distinti vengono rappresentati nel gruppo stesso).

L'idea alla base di Mondrian è quella di avere un raggruppamento ideale, in cui tutti i gruppi hanno la stessa dimensione, nel nostro caso k.

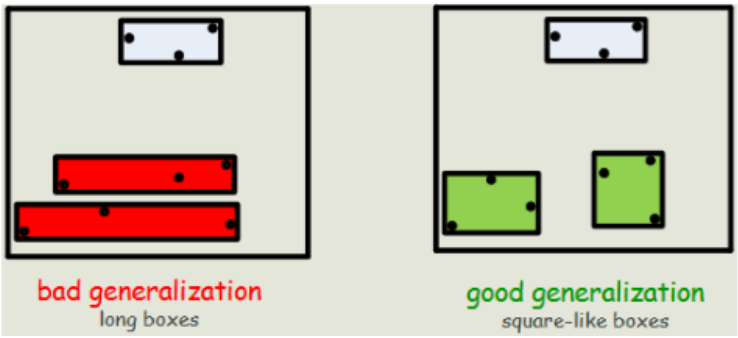
L'algoritmo quindi cerca di costruire gruppi di uguale dimensione.

Ipotizzando di lavorare su un dataset, l'algoritmo costruirà dei gruppi in base al livello di k-anonimity che si vuole avere andando a partizionare il dataset come possiamo vedere nell'esempio. Come detto prima, si cerca di partizionare e creare gruppi della stessa dimensione.



La qualità dei gruppi dipende dalla cardinalità del gruppo stesso, ovvero, in questo caso, ogni volta che viene creato un gruppo si cerca sempre di soddisfare la 2 k-anonimity.

Un buon gruppo contiene tuple con valori di QI simili. Tuttavia, tale proprietà non viene garantita da questo algoritmo. Per questo motivo viene definita una nuova metrica, ovvero la **NCP (Normalized Certainty Penalty)**, la quale misura il perimetro del gruppo per usarlo come penalty. Attraverso questa metrica possiamo vedere il vantaggio principale che apporta in quanto si vanno a raggruppare elementi simili tra loro.



Algoritmo Topdown

Fasi:

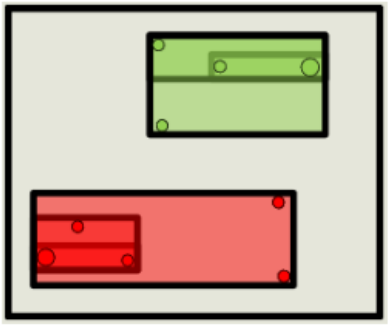
1. Si inizia con l'intero dataset;
2. Iterativamente si divide il dataset in 2;
3. Si continua fin quando non sono rimasti dei gruppi che contengono un numero meno di 2*k – 1 tuple.

Inizialmente trova i 2 punti più lontani (semi) che genereranno i 2 gruppi di split

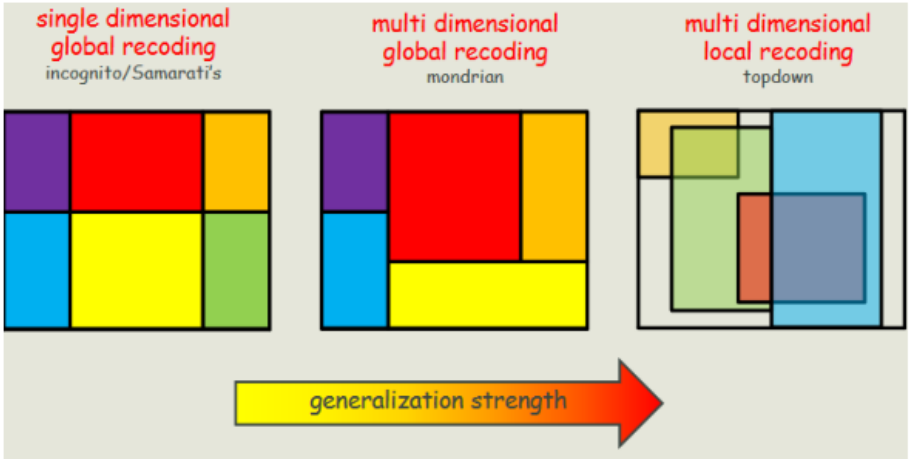
• ricerca quadratica euristica, non completa;

• i semi diventeranno casualmente i 2 punti divisi.

Va prima a trovare nell'unico spazio disponibile gli elementi con maggiore diversità, successivamente cerca gli elementi più vicini agli ultimi punti trovati e li aggrega in un gruppo in modo da costruire un perimetro. Quindi l'obiettivo è sempre quello di ingrandire i rettangoli cercando sempre di minimizzare il perimetro.



Generalization Strength



L-diversity

- Classe di equivalenza: L'insieme di tutte le tuple nella tabella generalizzata i cui valori non sensibili della tabella PT generalizzano in q*

- Lack of diversity: Quando il numero di tuple riguardanti un'attributo sensibile è molto minore di qualsiasi altro nella tabella (ad es. ci sono 100 tuple, 98 con diabette di tipo 1 e 2 con diabete di tipo 2)
- L-diversity: quando un blocco q* ha l>2 diversi valori sensibili in modo che i valori + frequenti nel q* block abbiano tutti la stessa frequenza
- Limiti: soggetto a attacchi basati sulla distribuzione dei valori interna ai blocchi (skewness attack e similarity attack), questi possono essere risolti grazie alla t-closeness

- **skewness attack**: succede quando la distribuzione in un blocco q è diversa dalla popolazione originale. Avendo quindi accesso a queste distribuzioni di dati diverse posso in qualche modo fare inferenza;
- **similarity attack**: si verifica quando un blocco q ha valori diversi ma semanticamente simili per l'attributo sensibile. (esempio di una tabella 2-anonimity ma con due attributi sensibili simili)

Figure 1: Attacchi alla l-diversity

T-closeness

- Un blocco q* ha t-closeness se la distanza tra la distribuzione di un attributo sensibile nella classe e nella tabella non supera una certa soglia t
- Una tabella si dice t-close se tutte le sue classi di equivalenza hanno t-closeness
- T-closeness è un parametro che consente di calibrare il trade-off privacy-utility

La t-closeness utilizza la Earth Mover Distance (EMD) invece che la distanza variazionale oppure la KL-distance perchè dobbiamo tenere conto le distanze di tipo semantico. La EMD misura la quantità di lavoro necessaria per trasformare una distribuzione in un'altra spostando la massa di distribuzione tra loro.

Problemi t-closeness:

- Soffre ancora del background knowledge attack
- Ogni attributo è un quasi-identifier nel giusto contesto e questo non possiamo controllarlo facilmente

delta-presence

Tenta di risolvere i problemi dati dalla membership disclosure impostando due parametri delta-min e delta-max tra la probabilità che un attaccante riesca a capire che una persona è all'interno della tabella privata dalla tabella generale.

Private Distributed Mining

Effettuare un mining locale sulle fonti di dati, ed utilizzare un aggregatore per mettere insieme i risultati ottenuti in un unico risultato. I dati poi vengono

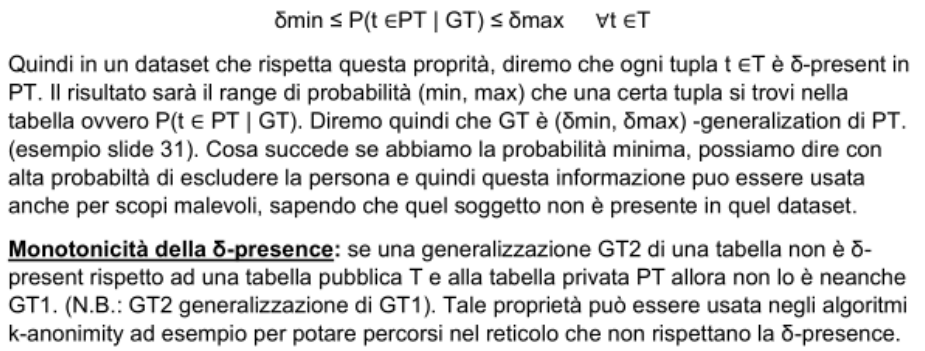


Figure 2: Formula e monotonicità della delta-presence

nascosti o generalizzati grazie ai seguenti metodi:

1. Data Obfuscation: obiettivo di nascondere l'informazione protetta –> modificando in maniera random i dati, scambiando valori tra record...
2. Summarization: rendere disponibili solo i *riassunti* dei dati
3. Data Separation: far rimanere i dati in chiaro in locale e i dati in remoto parziali e/o cifrati, eventualmente condividerli solo con una terza parte fidata

Data obfuscation

I dati originali vengono sporcati andando a modificare la sua distribuzione associata, così facendo si hanno dei nuovi valori che proteggono gli originali. Il processo consiste nel: dato originale –> randomizer –> dato sporco –> ricostruzione della distribuzione originale –> algoritmo di classificazione –> modello

Per ricostruire la distribuzione originale ricorriamo all'**original distribution reconstruction**: avendo valori distorti il nostro problema è quello di stimare la funzione di densità Fx dati i valori distorti wi e la funzione di densità Fy. Molto spesso utilizziamo una stima bayesiana per le funzioni di densità, che fa una stima probabilistica della funzione originale, rendendo questa stima quando più vicina all'originale possibile.

Utilizziamo il **bootstrapping method** (che utilizza il resampling del dato in input e fa inferenza sul dato dopo il resampling, ovvero resampled == sample) riuscendo effettivamente a stimare la funzione con l'utilizzo della appena citata con buona confidenza. Unico problema è la complessità dell'operatore, che include un integrale al denominatore, quindi si tratta di una **stima costosa**. Ci fermiamo utilizzando il **test del chi-quadro** effettivamente verificando che i dati siano distribuiti in maniera similare.

Costruiamo **alberi decisionali** per classificare dataset: Fase di crescita (dati partizionati per attributo, best split, e ripetiamo su ciascun nodo se i punti non sono della stessa classe) e fase di Pruning (generalizzazione per evitare *overfitting*, rimozione del rumore statistico)

Come vengono ricostruite le distribuzioni?

- Global: ricostruiamo per ogni attributo una volta all'inizio, costruiamo l'albero di decisione utilizzando i dati ricostruiti
- ByClass: prima dividiamo i dati d'addestramento, ricostruiamo ogni classe separatamente, costruiamo l'albero di decisione usando i dati ricostruiti
- Local: dividiamo i dati d'addestramento, ricostruiamo ogni classe separatamente, ricostruiamo su ciascun nodo mentre costruiamo l'albero

Dopo la ricostruzione utilizziamo la **privacy metric** che ci dice con una confidenza c che i dati sono compresi tra x1 e x2.

Data Separation

Utilizziamo la **Secure Multiparty Computation** che a sua volta utilizza il protocollo di **Oblivious Transfer**: si tratta di un modello semi-onesto (tiene traccia dei suoi calcoli intermedi), solo a due parti trusted, calcola un'approssimazione ID3-delta al posto del classificatore reale, ovvero una delta approssimazione di ID3, dove delta ha implicazioni sull'efficienza. *Funziona solo su attributi categorici*

ID3-delta

1. L'albero viene costruito dalla sola radice, a cui vengono assegnate tutte le istanze di addestramento
2. Scegliamo un attributo guardando l'entropia, che deve essere minimizzata
3. Creiamo tanti nodi quanti sono i possibili valori dell'attributo scelto
4. Ricorsivamente si usano i nodi come nuove radici, e così via

Privacy preserving ID3: abbiamo 3 parametri (R set di attributi, C attributo di classe, T set di transazioni, ovvero tuple descritte dall'insieme R)

Privacy Preserving ID3

Step 1 (si associa la var di class al nodo foglia): Se R è vuoto, restituisce un nodo foglia con il valore di classe assegnato alla maggior parte delle transazioni in T, quindi significa che se arrivo al nodo foglia posso fare la predizione.

Da considerare che il set di attributi è pubblico, quindi entrambe le parti sanno se R è vuoto.

Invece non sono note le porzioni del dataset: bisogna eseguire il protocollo Yao in cui viene usato l'approccio SMC per calcolare le seguenti funzionalità:

- Input (| T1 (c1) |, ..., | T1 (cL) |), (| T2 (c1) |, ..., | T2 (cL) |) che rappresenta la cardinalità delle tuple T1, T2 della classe c1 fino a cL.
- Output i (var di classe) dove | T1 (ci) | + | T2 (ci) | è il più grande;

Step 2: Se T è costituito da transazioni che hanno lo stesso valore c per l'attributo di classe, restituisce un nodo foglia con il valore c.

- I nodi foglia con più di una classe (nel set di transazioni), sono rappresentati con un simbolo fisso diverso da ci,
- Si forzano le parti a inserire il simbolo fisso o ci
- Controllare l'uguaglianza per decidere se nel nodo foglia prevale la classe ci
- Approcci vari per il controllo dell'uguaglianza (anche SMC per preservare la privacy di alcuni dati).

Step 3: (a) determina l'attributo che classifica meglio le transazioni in T cioè l'attributo da usare per lo split: ipotizziamo sia A (essenzialmente calcolato x * (ln x)).

(b, c) Chiamare in modo ricorsivo ID3δ (cioè la sua approx) per gli attributi rimanenti sugli insiemi di transazioni T (a1), ..., T (am) dove a1, ..., am sono i valori dell'attributo A. Poiché i risultati dello step 3 (a) e il i valori degli attributi sono pubblici, entrambe le parti possono partizionare individualmente il database e preparare i propri input per le chiamate ricorsive.

Differential Privacy

Paradigma che garantisce che vengano raggiunte le stesse conclusioni da una query indipendentemente dalla presenza o meno dell'individuo nel database.

Immagina di avere due database praticamente identici, uno con la tua informazione e uno senza. La differential privacy assicura che la probabilità che una query statistica produca un risultato è la stessa sia sul primo che sul secondo. Il fatto che tu ci sia o meno non produce un effetto significativo sull'output di questa query.

Un esempio molto semplice di applicazione è l'algoritmo randomizzato, che associa ad ogni persona in un set una probabilità random.

- **epsilon differential privacy**: la probabilità che la risposta ottenuta dal meccanismo provenga da D1 diviso la probabilità che provenga da D2 è minore o uguale a eˆepsilon. Devo trovare epsilon in modo che il rapporto sia 1. Se l'algoritmo rispetta la eps-dp allora la distanza tra le due distribuzioni sarà al massimo epsilon.

Risposta randomizzata Esempio base
Consideriamo un semplice meccanismo di risposta randomizzato
• Ai partecipanti allo studio viene richiesto di segnalare se hanno o meno una determinata proprietà P e di rispondere come segue:

- Lanciare una moneta
- Se esce croce, rispondi in modo veritiero
- Se esce testa, lancia una seconda moneta e rispondi "Sì" se esce testa e "No" se la coda

La versione della risposta randomizzata sopra descritta è ln (3) -differenzialmente privata
Proof:
Perdiamo il caso che Pr [Risposta = Sì | Verità = Sì] = ¾ (sarebbe 1/2+(1/2*1/2))
In particolare, quando la verità è "Sì" il risultato sarà "Sì" se la prima moneta esce croce (probabilità 1/2) o la prima e la seconda testa si presentano (probabilità ¼))

Figure 3: Risposta randomizzata, esempio

- **epsilon-delta differential privacy**: per ogni coppia di database vicini D1,D2 è estremamente improbabile che il valore osservato dal meccanismo D1 sia molto (più o meno) probabile che venga generato quando il database è D1 piuttosto a quando il database è D2. delta è, in questo caso, il grado di incertezza introdotto.

$$\Pr[\mathcal{A}(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(D_2) \in S], +$$

Figure 4: epsilon-delta

La differential privacy *non garantisce*:

1. Che i segreti rimangano segreti, in generale
2. Che delle conclusioni possano riflettere informazioni statistiche sulla popolazione
3. Che ciò che è pubblicamente osservabile (tu fumi), venga nascosto

Definizioni:

- **Sensitivity** di una funzione: calcola quanto vale la differenza di una certa funzione su un DB quando calcolo questa fuznione su database che differiscono di una sola tupla. Il massimo di questa variazione è la nostra sensitivity DeltaF. Questo valore rappresenta un upper-bound su quanto perturbare un input per preservare la privacy.
- **Meccanismo di Laplace**: Andiamo a sporcare una funzione f(D) con delle Y che sono variabili random tratte dalla laplaciana Lap(DeltaF/epsilon). Il meccanismo è applicabile solo su query numeriche e solamente quando sporcare la funzione non porta ad una riduzione drastica dell'utilità.
- **Meccanismo esponenziale**: Obiettivo di far ritornare una risposta precisa *senza* sporcare l'ouput. L'idea è di selezionare il miglior elemento da un set preservando però la differential privacy. Il miglior elemento è selezionato

grazie ad una funzione di utilità U. Il meccanismo provvede a mantenere la DP andando ad *approssimare* il massimo dello score dell'elemento. In altre parole alle volte è possibile che non sia veramente il massimo quello che viene ritornato. L'output dell'esponenziale ha la proprietà di essere sempre membro del set di output possibili, e questo è fondamentale quando sporcare una distribuzione comporterebbe all'inutilità del dataset (calendario di un evento ad esempio).

Sensitivity

- Globale: quella solita (non va bene in casi come quello della mediana)
- Locale: quanto varia al massimo una funzione se la calcolo su due sub-set del dataset

Meccanismi di LS:

1. Propose test release: usare un valore compreso tra la local sensitivity e la global sensitivity soddisfando la DP senza rivelare LS
2. Privately bounding LS: calcolare una stima privata di LS senza dare LS stesso
3. Smooth Sensitivity: utilizzare un nuovo parametro di sensitivity per creare un framework chiamato *Sample and Aggregate*. Ovvero suddividere un dato x in k sotto-campioni e calcolare f su ogni sub-valore e aggregarlo dopo con una funzione (es: mediana) utilizzando la DP solo in questo passaggio.

Esempi differential privacy

Query di conteggio

Esempio: quanti nel DB sono femmine?

- Sensitivity: 1
- Privacy epsilon-differenziale ottenuta con rumore: Lap(1/epsilon)
- Errore previsto: 1/epsilon

Per conteggi multipli m/epsilon invece che 1

Query di istogramma

Esempio: quanti nel DB ricadono nella categoria X?

- Sensitivity: 1
- Privacy epsilon-differenziale ottenuta con rumore: Lap(1/epsilon)

Query di media

- Sensitivity: [beta-alpha]/n
- Privacy epsilon-differenziale ottenuta con rumore: Lap([beta-alpha]/n * epsilon)

Classificatore Lineare

Trovare un vettore w che separa positivi e negativi, e trovare i parametri giusti in modo che il vettore separi bene le due classi. Utilizziamo l'empirical risk minimization per ottenere un risultato privato sfruttando esempi di un classificatore lineare.