Nom: Prénom:

M2 ISL. Sécurité avancée
Protection de la vie privée.

Aucun document n'est autorisé. Toutes les réponses doivent être justifiées. **Sans justification, une réponse est considérée comme fausse.**

1 Anonymisation syntaxique

Dans toute cette partie, on considère le jeu de données de la FIGURE 1a. Cette table contient 12 enregistrements précisant les attributs tels que

— *Nom* est l'identifiant,
— *Genre*, *Département* (d'enseignement) et *Statut* sont les quasi-identifiants ;
— *Salaire* est l'attribut sensible.

On souhaite publier des indicateurs par **département d'enseignement** tout en protégeant la vie privée des personnes dans cette base de données.

On considère les hiérarchie de généralisation suivante :

— *Genre* :
  — *Niveau 0* : H, F
  — *Niveau 1* : *
— *Département* :
  — *Niveau 0* : Info., Maths, Stats
  — *Niveau 1* : *
— *Statut* :
  — *Niveau 0* : Etud., Prof., Admin.
  — *Niveau 1* : *

1. Combien de possibilités de généralisation existe-t-il ?

2. L'outil ARX a proposé le treillis de généralisations donné à la FIGURE 1b. Expliquer le sens de celui-ci. Expliquer notamment le sens des nœuds, pourquoi (1,0,0) est au même niveau que (0,0,1), pourquoi (1,1,1) est au dessus de tous...

3. La FIGURE 2 présente les trois versions 2-anonymes correspondantes aux nœuds de ce treillis. Montrer que chacune de ces tables est 2-anonyme et associer à chaque table le nœud correspondant.

Nom	Genre	Département	Statut	Salaire ($K)
Adams	F	Info.	Etud.	10
Baker	H	Maths.	Prof.	60
Cook	F	Maths.	Prof.	100
Dodd	F	Info.	Admin	38
Engel	H	Stats.	Prof.	72
Flynn	F	Stats.	Prof.	88
Grady	H	Info.	Admin	40
Hayes	H	Maths.	Prof.	72
Irons	F	Stats.	Etud.	12
Jones	H	Stats.	Etud.	15
Knapp	F	Maths.	Prof.	100
Lord	H	Info.	Etud.	10

(a) Données concernant une université fictive

(b) Treillis de généralisations

FIGURE 1 – Mise en place de l'anonymisation

Genre	Département	Statut	Salaire ($K)
F	Info.	*	10
F	Info.	*	38
H	Info.	*	40
H	Info.	*	10
F	Maths.	*	100
F	Maths.	*	100
H	Maths.	*	60
H	Maths.	*	72
F	Stats.	*	88
F	Stats.	*	12
H	Stats.	*	72
H	Stats.	*	15

(a) Table A

Genre	Département	Statut	Salaire ($K)
*	*	*	10
*	*	*	60
*	*	*	100
*	*	*	38
*	*	*	72
*	*	*	88
*	*	*	40
*	*	*	72
*	*	*	12
*	*	*	15
*	*	*	100
*	*	*	10

(b) Table B

Genre	Département	Statut	Salaire ($K)
*	Info.	Admin	40
*	Info.	Admin	38
*	Info.	Etud.	10
*	Info.	Etud.	10
*	Maths.	Prof.	60
*	Maths.	Prof.	100
*	Maths.	Prof.	72
*	Maths.	Prof.	100
*	Stats.	Prof.	72
*	Stats.	Prof.	88
*	Stats.	Etud.	12
*	Stats.	Etud.	15

(c) Table C

FIGURE 2 – Plusieurs versions 2-anonymes

4. Démontrer que la version 2-anonyme qui correspond au nœud (1,1,1) a une valeur de Loss égale à 1.

5. Évaluer la valeur de valeur de Loss de la généralisation qui a abouti à la version 2-anonymes correspondent au nœud (1,0,0).

6. En terme d'homogénéité de données sensibles, montrer que la Table A et la Table C sont problématiques.

7. Quelles sont les tables 2-diverses et lesquelles ne le sont pas.

2 Confidentialité différentielle

2.1 Publication de données

8. Tout en respectant de la vie privée, on souhaite publier l'histogramme des emplois occupés par les personnes présentes dans une base de données (comme celle du recensement américain). Expliquer toute la démarche.

2.2 Apprentissage machine

9. Ci-dessous un extrait du code de la classe `Kmeans` de la bibliothèque `diffprivlib`. L'objectif est de l'expliquer en s'appuyant sur ce qui a été vu en cours.

```
1  def _update_centers(self, X, centers, labels, dims, total_iters):
2      epsilon_0, epsilon_i = self._split_epsilon(dims, total_iters)
3      geometric_mech = GeometricFolded(epsilon=epsilon_0, sensitivity=1, lower=0.5, upper=float("inf"))
4
5      for cluster in range(self.n_clusters):
6          cluster_count = sum(labels == cluster)
7          noisy_count = geometric_mech.randomise(cluster_count)
8
9          cluster_sum = np.sum(X[labels == cluster], axis=0)
10         noisy_sum = np.zeros_like(cluster_sum)
11
12         for i in range(dims):
13             laplace_mech = LaplaceBoundedDomain(epsilon=epsilon_i,
14                                                 sensitivity=self.bounds[1][i] - self.bounds[0][i],
15                                                 lower=noisy_count * self.bounds[0][i],
16                                                 upper=noisy_count * self.bounds[1][i])
17             noisy_sum[i] = laplace_mech.randomise(cluster_sum[i])
18
19         centers[cluster, :] = noisy_sum / noisy_count
20     return centers
21
22 def _calc_iters(self, n_dims, n_samples, rho=0.225):
23     epsilon_m = np.sqrt(500 * (self.n_clusters ** 3) / (n_samples ** 2) *
24                         (n_dims + np.cbrt(4 * n_dims * (rho ** 2))) ** 3)
25     iters = max(min(self.epsilon / epsilon_m, 7), 2)
26     return int(iters)
27
28
29  def fit(self, X, y=None, sample_weight=None):
30      iters = self._calc_iters(n_dims, n_samples)
31      centers = self._init_centers(n_dims)
32      labels = None
33      distances = None
34
35      # Run _update_centers first to ensure consistency of `labels` and `centers`, since convergence unlikely
36      for _ in range(-1, iters):
37          if labels is not None:
38              centers = self._update_centers(X, centers=centers, labels=labels, dims=n_dims, total_iters=iters)
39
40          distances, labels = self._distances_labels(X, centers)
41
42      self.cluster_centers_ = centers
43      self.labels_ = labels
44      self.inertia_ = distances[np.arange(len(labels)), labels].sum()
45      self.n_iter_ = iters
46
47      self.accountant.spend(self.epsilon, 0)
48      return self
```

(a) Dans la méthode `fit` à la ligne 30, pourquoi invoquer la méthode `calc_iters` ?

(b) Dans la méthode `fit` à la ligne 31, pourquoi invoquer la méthode `_init_centers` (non présentée) ?

(c) Dans la méthode `_update_centers` expliquer la ligne 7. Pourquoi avoir choisi un mécanisme géométrique dans ce contexte ?

(d) Dans la méthode `_update_centers` expliquer les ligne 17 et 19.