Examen 'IA et Predictive Analytics'

Instructions :

- Copier/coller ce texte dans un mail adressé à thomas.baudel@esiee.fr à la fin de l'examen, et remplir les réponses en dessous de chaque question.
- 5 à 6 lignes de texte par question sont généralement suffisantes pour obtenir une bonne note. Des points supplémentaires sont attribués pour des réponses plus détaillées. Chaque question apporte 2 points. Il n'est pas nécessaire de répondre aux questions marquées [BONUS] pour obtenir la note maximale mais des réponses justes à ces questions rapportent des points supplémentaires. Certaines questions sont beaucoup plus faciles que d'autres, et elles ne sont pas (toutes) par ordre de difficulté croissante...
- **Lorsque vous utilisez une réponse trouvée sur internet, donner l'hyperlien des sources utilisées.**

Barbosa Vaz, Vincent

Conduite de projets en science des données

1: IA
a) Décrire la différence entre approche réaliste et approche utilitariste dans la démarche scientifique.

L'utilitarisme est une doctrine en philosophie politique ou en éthique sociale qui prescrit d'agir (ou de ne pas agir) de manière à maximiser le bien-être collectif, entendu comme la somme ou la moyenne de bien-être (bien-être agrégé) de l'ensemble des êtres sensibles et affectés. - https://fr.wikipedia.org/wiki/Utilitarisme

Issue de la philosophie réaliste des sciences, l'approche réaliste postule en effet une causalité générative selon laquelle toute action produit des effets par l'interaction du contexte et de mécanismes. Elle repose sur plusieurs concepts clés : causalité générative, mécanismes, configuration Contexte-Mécanisme-Effet et demi-régularités. Cette approche a été popularisée par le sociologue britannique Ray Pawson, qui œuvre dans le domaine de l'évaluation d'interventions sociales et dont l'objectif consiste à changer la pratique, le comportement, la compréhension ou la perception de personnes, groupes ou organisations qui en sont les cibles. - http://unitesoutiensrapqc.ca/lapproche-realiste-en-recherche-axee-sur-le-patient/

L’approche utilitariste tente de satisfaire le bien commun, l’approche réaliste est une pensée “pragmatique” qui va répondre à des besoins moins “louables” mais dont la réalisation est faisable dans un moindre effort (économique par exemple).

b) Watson Studio: décrire une caractéristique importante de Watson Studio comme environnement de développement en science des données, qui le rend utile en situation de développement en entreprise.

Watson Studio agrège dans une même plateforme de développement plusieurs services qui répondent au besoin de l’ingénieur en science des données. De l'exploration de données à la mise à l’échelle en passant par le développement de modèles, l’utilisateur pourra travailler dans un même environnement au cours des étapes clés d’un projet en data science. Rassembler dans une même plateforme ces outils représentent un gain de temps considérable pour l’utilisateur mais aussi sur la formation de ses employés qui utilisent un unique environnement de travail. C’est la force de Watson Studio.

2: programmation logique/chainage avant
Dans un langage à base de règles simple en chainage avant (on appelle cela un système de production) on a le programme:

var *input[]=[], result[]=[], i=1, tmp=0;*

when *input.length>0* ***and*** *i >= input.length* ***then*** *result.append(input[tmp]), input.removeAt(tmp), i=1, tmp=0;*
when *input[i] < input[tmp]* ***then*** *tmp=i, i=i+1;*
when *input[i] >= input[tmp]* ***then*** *i=i+1;*

Lorsque l'instruction *input=[2,0,5,4,9];* est exécutée, que contiendront les variables *result* et *input* en retour? Expliquer l'algorithme.

result = [0, 2, 4, 5, 9]
input = []

L’algorithme trie la liste en entrée.

3: Smart City: Trouver sur internet 3 logiciels commerciaux **professionnels** destinés à remplir un rôle similaire à celui du projet SmartDeliveries. En vous basant sur la présentation commerciale, identifiez leurs principales caractéristiques démarquantes (quels fonctionnalités mettent-ils particulièrement en avant par rapport à la concurrence). Fournir les références utilisées.

City Logistics

Objectif : réduire les nuisances associées au transport de fret dans les zones urbaines tout en supportant le développement économique et social des villes.

GeoConcept
Propose une application et le marketing est accés sur la rentabilité.

https://fr.geoconcept.com/app-plan-tournees-cloud

Optilogistic

Prend en compte les contraintes métiers et organisationnelles.

http://www.optilogistic.fr/

4: trafic routier
a- Quelles sont les principales variables mesurées par un détecteur de trafic?

Le nombre de véhicule circulant sur un tronçon par unité de temps (pour en déduire la congestion). La vitesse des véhicules.

Les principales variables sont le flux et la densité du trafic.

b- Qu'est ce que le diagramme fondamental d'un détecteur de trafic, pourquoi est-il utile pour mesurer et prévoir la congestion?

Le diagramme fondamental d'un détecteur de trafic traduit une relation entre le flux du trafic (nombre de véhicules/heure) et la densité du trafic (nombre de véhicules/km). Il permet de prédire la capacité d'un système routier à absorber de manière efficace ou non une augmentation du trafic. Il permet de donner des informations importantes à l'ingénieur routier afin de minimiser la congestion par régulation.

c- quel est le débit typique maximal d'un tronçon à une voie
- en zone urbaine
- sur voie rapide ou autoroute
- Pourquoi cette différence?

Si on considère une autoroute à trois voies, le débit maximal ou la capacité sera typiquement de 6 000 à 7 000 véh/h. (source : Comprendre le trafic routier, Christine Buisson et Jean-Baptiste Lesort). Le débit typique maximal pour un tronçon à une voie sur voie rapide est de 3000 à 4000 véh/h (source : tracé des diagrammes fondamentales en TP).

Pour un tronçon à une voie le débit typique maximal est de 1000 à 2000 véh/h. (source : tracé des diagrammes fondamentales en TP).

Cela s'explique par la vitesse de circulation qui est plus élevée sur voie rapide (attention à l'effet accordéon à grande vitesse). Les voies rapides ne sont pas sujettes aux feux de circulation ni aux ralentissements des villes dues à leur fonctionnement (chargement/déchargement de livraisons, piétons, services municipaux, croisements, virages forts etc.).

5: temps de parcours

a) Quelles sont les principales variables prédictives du temps de parcours d'un camion de livraison en ville, par ordre d'importance décroissante? (déterminées en cours)

Méthode : TP3 lancer le programme decision_tree.py du dossier "temps appris".

Sortie du programme :

```
duration 0.006135741453721717
time 0.0
dayOfWeek 0.00282424012286369
x1 0.006039584962792454
y1 0.023583972627551495
x2 0.005640101217904607
y2 0.005750593222739783
distanceOrigine 0.014036259238967288
distanceDestination 0.011354709264904754
angleOrigine 0.007202142777827465
angleChemin 0.0013995966602088265
averageCategory 0.0001228289711182354
lanes 0.0016919992708233165
distance 0.7053753655435705
staticDuration 0.014164304372882968
missionName 0.1946785602921229
```

Les principales variables sont :

- la distance
- durée statique
- le jour de la semaine

b) Citer 2 facteurs potentiels affectant les temps de parcours et difficiles à mesurer avec les données fournies.

Deux facteurs potentiels sont : la période de vacances scolaires ou le nombre de feux de circulation.

6: géoréférencement
On a un fichier d'adresses tel que vu en cours:
ID, BASE, KIND, CITY, FROM, TO
1, Docteur Bouchut, Rue du, Lyon, 1, 100
etc...
a) Décrire la logique d'une fonction python qui permet de retrouver l'identifiant de tronçon (ID) correspondant à chaque adresse, en supposant que la base d'adresses est stockée dans un tableau, et que l'on a une fonction distance(a, b) qui retourne la distance de Levenshtein entre 2 chaines.

17, rue Bouchut, Lyon
17 rue du Docteur Bouchot, Lyon

La distance de levenshtein permet de mesurer le nombre de modifications nécessaires pour modifier une chaîne A en une chaîne B. On l'utilise pour trouver l'adresse la plus pertinente (la plus proche).

On aura pris soin de créer l'adresse complète en concaténant les colonnes nécessaires.

lev("17, rue Bouchut, Lyon", dataframe[nom de la rue]) qui nous donne la distance entre toutes les adresses et l'adresse en entrée. On sélectionne la plus faible valeur puis on retourne l'ID de cette ligne (adresse).

b) proposer cette fonction en python (améliorant celle vue en cours)

```
linkid(s, addresses):
        """ addresses contient une liste de tronçons [id, base, kind, city, from, to], s
l'adresse à trouver """
        return id

linkid(s, addresses):
```

levenstein[‘dist’] = levenstein(s, addresses[adresse_complete])

dist_sorted = sort(dist, on=addresses.adresse_complete, ascending=True)

id = dist_sorted[id][0]

return id

On suppose que adresse_complete contient l’adresse complète à comparer.

Prescriptive Analytics

Question 1.
Les affirmations suivantes sont-elles vraies ou fausses:

a- Un problème de décision est dans la classe de complexité NP si et seulement si il n'existe pas d'algorithme polynomial pour le résoudre.

Faux, La classe NP est formée des problèmes de décision qui peuvent être résolus par un algorithme polynomial non déterministe. On doit pouvoir vérifier « rapidement » (complexité polynomiale) si une solution candidate est bien solution.

b- Dans l'industrie, la majorité des problèmes d'ordonnancement sont résolus grâce à des heuristiques.

Faux, il existe plusieurs approches. Les heuristiques sont une des approches possibles.

c- Le problème suivant possède exactement trois solutions:

```
u in {1,3}
v in {1,2}
w in {3,4}
x in {1,5}
y in {4,5}
allDifferent(u,v,w,x,y)
```

Faux, il en possède 2 :
u=1, v=2, w=3, x=5, y=4
u=3, v=2, w=4, x=1, y=5

d- L'algorithme de résolution de CP-Optimizer est un algorithme exact: si un problème d'optimisation est faisable, il garantit de trouver une solution optimale.

Vrai ! Cependant il ne retourne pas forcément la solution optimale mais une solution “acceptable”. C’est à dire qu’on peut stopper l'exécution du programme à tout moment et se satisfaire de la solution proposée.

Question 2.

a- En cherchant sur internet, décrivez un problème d'optimisation combinatoire non vu dans le cours dont la version de décision est un problème NP-Complet.

Knapsack problem (en français : Problème du sac à dos).

b- Décrivez une petite instance particulière de ce problème d'optimisation (avec des valeurs pour chacune des données).

Objet	1	2	3	4
Poids	5	10	10	5
Valeur	10	15	10	20
Valeur/Poids	2	1.5	1	4

Taille du sac : 15

En notant V(i) la valeur du sac obtenue :

On met l’objet dans le sac, donc V(i) = V(i-1) + pi
On ne met pas l’objet dans le sac, donc V(i) = V(i-1)

On prend les objets 4 et 1 (valeur/poids = 6).

c- Donnez une solution faisable non-optimale et une solution optimale de cette petite instance.

Non-optimale : objets 1 et 2 (valeurs/poids = 3.5)
Optimale : objets 4 et 1 (valeur/poids = 6)

Question 3.

Deux principes fondamentaux de la Programmation par Contraintes sont (1) la recherche arborescente et (2) le filtrage du domaine des variables. Décrivez brièvement ces principes, leurs rôles et la façon dont ils sont mis en oeuvre durant la résolution.

Dans le cas de la résolution sur domaines finis, il est en théorie possible d'énumérer toutes les possibilités et de vérifier si elles violent ou non les contraintes, cette méthode est appelée générer et tester. Cependant, cela s'avère impraticable pour des problèmes de taille moyenne en raison du grand nombre de combinaisons possibles. Une des majeures parties de la résolution, appelée « filtrage », a pour but d'éviter cette énumération exhaustive. Elle consiste à déduire à partir des contraintes les valeurs impossibles. (source : https://fr.wikipedia.org/wiki/Programmation_par_contraintes#Recherche_arborescente)

Cette série de découpages du problème peut être représentée sous forme d'un arbre. Le but de la recherche est de parcourir cet arbre (en le construisant au fur et à mesure) jusqu'à trouver une solution au problème tandis que le filtrage consiste à « élaguer » cet arbre en supprimant toutes les parties n'aboutissant qu'à des contradictions. (source : https://fr.wikipedia.org/wiki/Programmation_par_contraintes#Recherche_arborescente)

On peut décrire la recherche arborescente comme suit : on teste successivement les valeurs possibles. Si une solution partielle ne vérifie pas toutes les contraintes, on revient à l'étape précédente. Afin d'optimiser le parcours de l'arbre on filtre le domaine des variables ce qui conduit à cherche une solution dans un ensemble des possibilités réduit.

Question 4.

Un problème classique en ordonnancement est le problème d'open-shop pour lequel un ensemble de n jobs doivent être exécuté sur m machines. Chaque job consiste en un ensemble de m opérations de durée connue, devant être exécutées dans un ordre arbitraire sur les machines. Plus précisément, si l'opération o_ij désigne la jème opération du job i, cette opération utilise la machine j et sa durée est D_ij. L'ordre des opérations du job i n'est pas connu d'avance: les m opérations o_ij d'un job i donné doivent être ordonnées mais cet ordre est libre. D'autre part, une

machine ne peut pas effectuer plus d'une opération à la fois. L'objectif est de déterminer les dates de début et de fin de chaque opération de manière à minimiser la date de fin du plan. Ce problème d'optimisation est NP-difficile.

Les données d'entrée du problème sont donc:
- n, le nombre de jobs
- m, le nombre de machines
- D_ij (i in [1,n], j in [1,m]), la durée de la jème opération du job i

a- Décrivez un modèle CP Optimizer à base de variables d'intervalles et de contraintes noOverlap pour résoudre le problème d'open-shop.

b- [BONUS] Décrivez un modèle CP Optimizer pour une variante du problème d'open-shop pour laquelle les machines peuvent effectuer au plus deux opérations en parallèle.