AU. : 2022/2023
Filière : Master GLCC, S2
Module : Blockchain & web3
Professeur : M. OUALLA

كلية العلوم – القنيطرة
Faculté des Sciences – Kenitra

Examen Blockchain & web3

1. Définir les termes suivants :

   Blockchain, consensus, application décentralisée, smart contrat.

2. Citez quelques différences entre web2.0 et web3.0
3. Quelle sont les différences entre l'architecture centralisée et décentralisée ?
4. Donner les points forts d'une fonction de hashage
5. Quel algorithme de hashage est à la base du bitcoin ?
6. Expliquez le principe du mécanisme de consensus POW
7. Quelles sont les étapes à suivre pour développer une dApp ?
8. Donnez les étapes pour déployer un smart contrat dans Truffle
9. Quelle est la limite principale de l'utilisation de la Blockchain locale Ganache ?
10. Combien de mots sont nécessaires pour retrouver le contenu de mon portefeuille Metamask égaré ?

Faculté des Sciences
كلية العلوم

Filière : Master Génie Logiciel pour le cloud

Module : Internet des Objets

Examen Final

Exercice 1 :

1. Définissez le concept d'un système embarqué.
2. Donner quatre caractéristique des systèmes embarqués en montrant leurs intérêts pour le choix d'un système embarqué optimale.
3. Expliquez brièvement ce qu'est l'Internet des Objets (IoT) et son importance dans le contexte actuel.
4. Donnez deux exemples d'applications pratiques de l'IoT dans des domaines tels que la santé, l'agriculture ou la logistique, en expliquant comment l'IoT améliore ces secteurs.
5. Décrivez les principaux défis de sécurité liés à l'IoT et proposez au moins deux mesures pour les surmonter.

Exercice 2 :

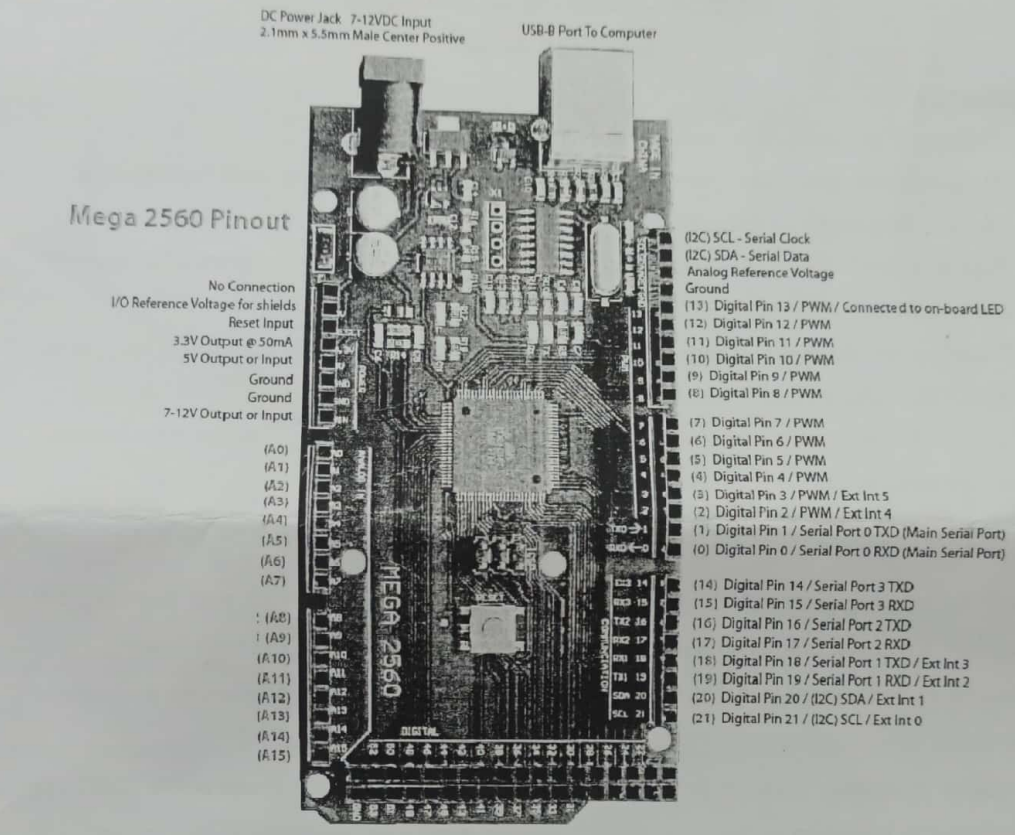
1. Écrivez un programme Arduino qui allume une LED connectée au port numérique 9 pendant 3 secondes, puis l'éteint pendant 2 secondes. Répétez ce cycle trois fois.
2. Expliquez brièvement la différence entre les fonctions setup() et loop() dans un programme Arduino. Comment sont-elles utilisées dans le contexte de l'exercice 2.1 ?
3. Utilisez le module Arduino pour lire les données d'un capteur de température (DHT11) Affichez la température mesurée sur le moniteur série avec une mise à jour toutes les 5 secondes.

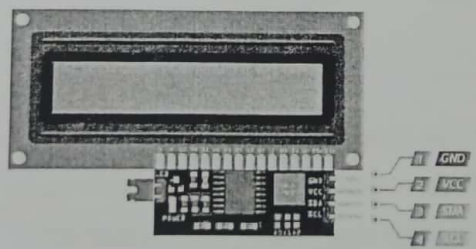
Exercice 3 :

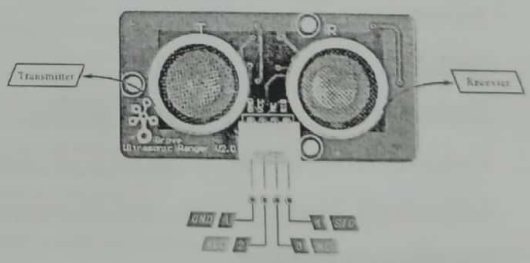
1. Expliquez en détail le principe de fonctionnement d'un capteur ultrasonique.
2. Concevez un schéma de connexion pour un capteur ultrasonique (HC-SR04) et un afficheur LCD (16x2). Identifiez clairement les broches utilisées sur chaque composant et expliquez leur rôle dans le circuit.
3. Écrivez un programme Arduino qui permet de mesurer la distance entre le capteur ultrasonique et un objet, puis affiche cette distance en centimètres sur l'afficheur LCD. Assurez-vous de commenter chaque section du code pour expliquer son fonctionnement.

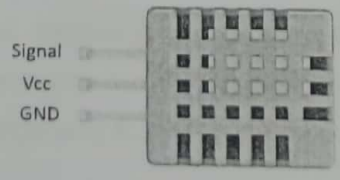
Faculté des Sciences
كلية العلوم

Annexe









Nom complet :

Note:

FACULTE DES SCIENCES
KENITRA

Examen Big Data – Master GLCC – S3 – Pr. Hatim DERROUZ

Questions de cours :

1. Décrivez en détail le fonctionnement de l'algorithme de NameNode et DataNode dans HDFS.

2. Décrivez les différents types de transformations et d'actions disponibles dans Spark RDD.

3. Décrivez les différentes bibliothèques disponibles pour Spark (Spark SQL, Spark Streaming, Spark MLlib).

4. Expliquez en brief un des Framework de Big Data

HBase est :
a) Une base de données relationnelle
b) Une base de données NoSQL
c) Un outil d'analyse de données
d) Un système de fichiers distribué
e) Un framework de machine learning

Laquelle des caractéristiques suivantes N'EST PAS une caractéristique du Big Data ?
a) Volume important
b) Vérité unique
c) Variété de formats
d) Vélocité de génération
e) Grande valeur potentielle

Lequel des composants suivants n'est PAS un composant central de Hadoop ?
a) HDFS
b) YARN
c) MapReduce
d) Hive
e) Spark

Quelle est la fonction principale de YARN dans Hadoop ?
a) Stocker des données massives
b) Exécuter des tâches de traitement de données
c) Gérer les nœuds du cluster
d) Fournir une interface SQL pour interroger les données
e) Toutes les réponses ci-dessus

Le modèle de programmation RDD de Spark repose sur :
a) Des instructions séquentielles
b) Des collections de données distribuées et immuables
c) Des requêtes SQL complexes
d) Des modèles d'apprentissage automatique pré-entraînés
e) La programmation parallèle avec des threads

Quelle est la fonction principale de Hive ?
a) Stocker des données Big Data
b) Exécuter des tâches de traitement de données en temps réel
c) Fournir une interface SQL pour interroger des données sur HDFS
d) Gérer les nœuds d'un cluster Hadoop
e) Analyser des images et des vidéos

Université Ibn Tofail
Faculté des Sciences
Département d'Informatique
Kénitra

A.U. 2022/2023
Master Génie Logiciel
pour le Cloud Computing
Semestre 1
Machine Learning

Epreuve (1h30)

Session de rattrapage

Le jeu de données de travail mettent en relation 50 attributs descriptifs d'écrivant des caractéristiques architecturales d'immeubles résidentiels (surface des murs, surface des toits, orientation, etc) et 2 attributs cibles : les charges de chauffage (Yheat) et les charges de climatisation (Ycool) de ces immeubles. Le problème d'apprentissage automatique consiste à prédire les attributs cibles, c'est-à-dire les charges, en fonction des attributs descriptifs.

Les données ont été recueillies pour 2000 immeubles et stockées dans un fichier data.csv. Tous les attributs descriptifs sont numériques. Pour les charger en Python, on utilisera le code suivant:

```
import numpy as np
data = np.loadtxt("./data.csv")
X = data[:,:-2]
Y = data[:,-2:]
Yheat = Y[:,0]
Ycool = Y[:,1]
```

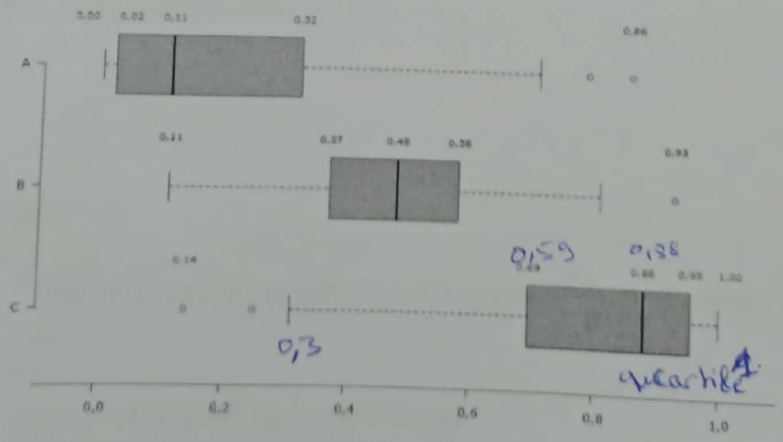
I. Prétraitements des données :

Dans la phase des prétraitements des données (Data Pre-processing), on utilise des mesures de dispersion en particulier la boîte à moustaches qui est déterminée par l'étendue E, la médiane M, le quartile inférieur Q1, le quartile supérieur Q3 et l'écart interquartile IQ = Q3 – Q1.

1. Déterminez ces mesures de l'ensemble de données suivant :

6, 47, 49, 15, 43, 41, 7, 39, 43, 41, 36

2. Interprétez la distribution des données C en donnant les 5 mesures.



```
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
a = metrics.accuracy_score (y_test, y_pred)
kfold = KFold(n_splits=5, random_state=25, shuffle=True)
results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
knnclass_cv=model_selection.GridSearchCV( model, params, cv=5, scoring='accuracy')

k_values = [5, 7, 9, 13, 15, 17, 19, 21]
params={'n_neighbors': k_values}
knnclass_cv.best_params_['n_neighbors']
knnclass_cv.best_score_
y_pred = knnclass_cv.best_estimator_.predict(X_test)

km = cluster.KMeans(n_clusters=k)
dbscan = cluster.DBSCAN(eps=0.1, min_samples=2)
cha = cluster.AgglomerativeClustering(linkage="ward", distance_threshold = 6, n_clusters=None)
km.fit(X)
cs= metrics.silhouette_score(X, km.labels_)
ir=metrics.adjusted_rand_score(labels, km.labels_)
pca = PCA(n_components=5)
```

```
target = np.loadtxt('./labels', dtype='int')
```

Nous allons comparer plusieurs méthodes d'apprentissage supervisé :

1. les k-plus proches voisins

   model1 = KNeighborsClassifier(n_neighbors= k)

   Hyperparamètre à régler : k.

2. SVM à noyau rbf

   Model3 = SVC(gamma= g, kernel = 'rbf')

   Hyperparamètre à régler : g.

Ecrivez les codes permettant de :

(a) séparer les données de travail en des données d'entrainement et de test,

(b) sélectionner les hyperparamètres des algorithmes d'apprentissage sur l'échantillon d'entrainement par validation croisée,

(c) afficher le score accuracy de ces algorithmes évalués sur l'échantillon test.

ANNEXE

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn. neighbors import KNeighborsClassifier
knnclass = KNeighborsClassifier(n_neighbors=7)
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn import tree
from sklearn import metrics
from sklearn import cluster
from sklearn.decomposition import PCA

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=25,stratify=y)
scaler = StandardScaler().fit(X_train)
X_train =scaler.transform(X_train)
rmse= metrics.mean_squared_error(y_test, y_pred, squared=False)))
Le coefficient de détermination : R2 = metrics.r2_score(y_test, y_pred)
```

3. Les données brut d'un jeu de données qui servent à l'apprentissage automatique peuvent avoir des valeurs diverses. Afin de normaliser les données, on utilise des transformations.

a- Quel est l'objectif de ces transformations ?

b- Les types de transformations des données les plus utilisés sont :

- Mise à l'échélle (MinMaxScaler)
- Standarisation des données (StandardScaler)
- Normalisation des données (Normalizer)

Donnez l'interprétation de chaque transformation sur une distribution de données Y=(Yi).

II. Réduction de la dimension

Notre jeu de données **X = data[:,:-2]** contient 50 attributs descriptifs, on veut réduire la dimension du jeu de données X par l'analyse en composante principale (ACP).

(a) Quel est l'objectif de la réduction de la dimension ?

(b) Appliquer l'ACP à 10 composantes principales afin de réduire la dimension des données X.

(c) Effectuer les projections des données X sur les deux premiers axes principaux.

III. Clustering :

Par les méthodes de clustering, on peut répartir le jeu de données en plusieurs classes selon les charges de chauffage et de climatisation (**Yheat**, **Ycool**). On va utiliser et comparer les deux méthodes suivantes :

- K-means
- DBSCAN

(a) Décrire en bref les avantages et inconvénients de chaque méthode.

(b) Ecrire le code pour chacune des méthodes citées.

(c) Quels sont les indices ou coefficients à utiliser pour évaluer une méthode de clustering ? Comment peut-on faire pour déterminer le choix optimal des hyperparamètres ?

IV. Problème de classification

On peut transformer le problème initial en un problème de classification. Par une méthode de clustering, on peut répartir les charges de chauffage et de climatisation en 3 classes : **faibles, moyennes, élevées** codées 0, 1, 2 respectivement.

Les étiquettes des classes se trouvent dans le fichier "**./labels**". Le code Python permettant de charger ce fichier :

Université Ibn Tofail
Faculté des Sciences
Département d'Informatique
Kénitra

Année universitaire : 2022/2023
Filière : MASTER GLCC
Semestre : 1
Session rattrapage

Examen du module : Python

Exercice 1 :

Supposons que vous travaillez pour une entreprise qui vend des produits dans différentes régions. Vous avez reçu un fichier CSV contenant les ventes mensuelles de ces produits pour chaque région. Votre tâche est de créer un programme Python qui lira les données à partir du fichier CSV, effectuera une analyse des ventes et affichera un graphe pour représenter les données.

Le format du fichier CSV est le suivant :

```
Region, Produit, Janvier, Février, Mars, Avril, Mai, Juin, Juillet, Août, Septembre, Octobre, Novembre, Décembre
Region1, Produit1,100,120,80,90,110,130,140,150,100,120,130,140
Region1, Produit2,50,60,70,80,90,100,110,120,130,140,150,160
Region2, Produit1,200,210,220,230,240,250,260,270,280,290,300,310
Region2, Produit2,300,310,320,330,340,350,360,370,380,390,400,410 ...
```

1. Créez une classe **SalesData** pour stocker les données de ventes pour chaque région. La classe devrait avoir les attributs suivants :
   - **region**: une chaîne de caractères représentant le nom de la région.
   - **produits** : un dictionnaire où les clés sont les noms des produits et les valeurs sont des listes représentant les ventes mensuelles pour chaque produit.
2. Implémentez une méthode **load_from_csv** dans la classe **SalesData** qui prendra le nom du fichier CSV en entrée et chargera les données de ventes à partir du fichier.
3. Implémentez une méthode **calculate_total_sales** dans la classe **SalesData** qui calculera les ventes totales pour chaque produit dans chaque région (la somme des ventes mensuelles).
4. Utilisez une structure de données appropriée pour stocker les instances de la classe **SalesData** pour chaque région.
5. Créez une fonction **plot_sales_data** en dehors de la classe **SalesData** qui prendra en entrée les instances de **SalesData** pour chaque région et tracera un graphe à barres représentant les ventes totales pour chaque produit dans chaque région. Assurez-vous de donner des noms appropriés aux axes et aux légendes du graphe.
6. Gérez les exceptions pour les cas où le fichier CSV n'est pas trouvé ou les données sont mal formatées.

Conseils :

- Vous pouvez utiliser le module **csv** pour lire les données à partir du fichier CSV.
- Pour la visualisation, utilisez le module **matplotlib**.

DictReader

Exercice 2 :

Créez un module Python personnalisé qui contient une fonction pour calculer la racine carrée d'un nombre en utilisant la méthode de votre choix. Importez ce module dans un nouveau script Python et utilisez cette fonction pour trouver les racines carrées de 5 nombres différents.

Faculté des Sciences
كلية العلوم

Année universitaire 2022/2023

Master Génie Logiciel pour le Cloud Computing
Examen de rattrapage du Module Noyau et Administration des Services Linux

Exercice 1(5 pts) : Donner les commandes Shell qui permettent de :

1- Rechercher tous les fichiers des systèmes se terminant avec la Chaîne « **sh** » ls . *sh
2- Chercher et Afficher en format long tous les fichiers appartenant à un utilisateur donné (choisissez un utilisateur qui existe dans votre système)
3- Chercher et Afficher en format long les fichiers de vote maison (HOME) datant plus de 20 jours
4- Chercher et Afficher en format long dans **/usr** les fichiers dont la taille dépasse 1Mo
5- Chercher et Afficher en format long dans **/bin** les fichiers dont les droits sont fixés à 755
6- Chercher dans le répertoire courant tout les fichiers qui commence par un caractère numérique et qui datent plus de 20 jours ls [0-9]*

Exercice 2(5 pts) : Donner les commandes Shell (Sous le compte du root) qui permettent de

1- Régler la date et l'horloge
2- Créer dans 5 minutes le répertoire $HOME/ATT1
3- Créer à 14h 30 min le fichier $HOME/At_fic
4- Enregistrer le taux d'occupation d'espace disque dans le fichier occup.log
5- Planifier le nettoyage de /tmp chaque vendredi à 20h

Exercice 3(5 pts)

Écrire un script qui convertit en minutes et secondes un temps exprimé en secondes (Le temps est passé en paramètre).

Exercice 4(5 pts)

Créer la commande **copier.** La commande reçoit en argument deux noms de fichiers, la source et la destination. Le script se termine et affiche un message d'erreur si l'une des conditions suivantes est réalisée :

- Le nombre d'arguments est incorrect.
- Le fichier source n'existe pas ou il n'est pas copiable (pas d'accès en lecture).
- Le fichier source n'est pas un fichier ordinaire.
- Le fichier destination existe.
- Le répertoire de destination n'est pas accessible en écriture.
- La copie a échoué.

Université Ibn Tofail
Faculté des Sciences
Département d'Informatique
Kénitra

Année universitaire : 2022/2023
Filière : MASTER GLCC
Semestre : 1
Session normale

Examen du module : Python

Exercice 1 :

Ecrire une classe **Words** dont la méthode **_init_** prend comme paramètre **un fichier texte**, et possède une méthode **stats**, renvoyant **la liste des couples (w,n)** des mots apparaissant dans le texte et de leur nombre d'occurrences. Cette méthode prendra un paramètre **ordre**, et renverra la liste triée par ordre alphabétique croissant sur **w** ou par ordre décroissant sur **n** selon que **ordre** vaudra 0 ou 1.

Représenter la liste des couples(w,n) par un graphique

Exercice 2 :

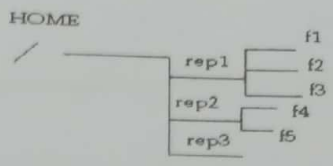
1. Créer une classe "**Etudiant**" avec les propriétés suivantes :
   - nom (str)
   - prénom (str)
   - âge (int)
   - notes (dict) : un dictionnaire qui contiendra les notes de l'étudiant sous la forme {matière: note}
2. Créer une sous-classe "**EtudiantBoursier**" qui hérite de la classe "Etudiant" et qui a une propriété supplémentaire :
   - bourse (float)
3. Ajouter une méthode à la classe "**Etudiant**" qui permet de calculer la moyenne de l'étudiant en prenant en compte toutes les matières. Si une matière n'a pas de note, elle sera considérée comme égale à 0.
4. Créer une classe "**Promotion**" qui contient une liste d'objets "**EtudiantBoursier**" et qui a les méthodes suivantes :
   - **ajout_etudiant(self, etudiant: EtudiantBoursier)** : ajoute un étudiant à la liste
   - **suppression_etudiant(self, nom: str, prenom: str)** : supprime l'étudiant de la liste en utilisant son nom et son prénom comme clés
   - **moyenne_promo(self)** : calcule la moyenne de la promotion en prenant en compte toutes les matières de tous les étudiants

- **graphique_moyennes(self)** : affiche un graphique des moyennes de tous les étudiants de la promotion. Le graphique devra afficher le nom de chaque étudiant sur l'axe des x et sa moyenne sur l'axe des y.

5. Pour lire les données des étudiants à partir d'un fichier JSON, vous devrez créer une méthode pour charger les données à partir du fichier JSON et créer des objets "**EtudiantBoursier**" à partir des données lues.
6. Gérer les exceptions lorsque des données incorrectes sont fournies (par exemple, un âge négatif ou une note non comprise entre 0 et 20).

Master Génie Logiciel pour le Cloud Computing
Examen du Module Noyau et Administration des Services Linux

Exercice 1(5 pts) : Donner les commandes Shell qui permettent de



1. Donner la commande Shell qui permet de créer l'arborescence suivante :
2. Mettre-vous au répertoire racine de cette arborescence :
   a. Déplacer f1 au rep3
   b. Renommer f2 par fichier (mv)
   c. Mettre une deuxième copie f4 dans le rep3 avec le nom f77
   d. Supprimer rep2

Exercice 2(5 pts) : Donner les commandes Shell (Sous le compte du root) qui permettent de

1. Créer un dossier /service/projet
2. Créer un nouveau groupe Projet
3. Créer les membres de ce groupe : user1, user2, user3
4. Mettre Projet est le groupe du dossier /sevice/projet
5. Vérifier les droits attribués par défaut au propriétaire et au groupe de projet.
6. Attribuer à /service/projet les privilèges de root.
7. Sous le compte user1 créer un fichier donnee.
8. Les droits d'accès du fichier donnee sont limités au propriétaire
9. Vérifier l'accès des autres utilisateurs (root, user2 et user2)

Exercice 3(5 pts)

Ecrire un scripte shell qui permet de trouver tous les fichiers d'extension java dans l'arborescence courante qui contiennent la chaine de caractères passée en paramètre

Exercice 4(5 pts)

On donne les résultats de 3 commandes netstat ci-dessous, extraites de la même machine :

```
$ netstat -nr
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
198.5.203.0 0.0.0.0 255.255.255.0 U 1500 0 0 eth0
127.0.0.0 0.0.0.0 255.0.0.0 U 3584 0 0 lo
0.0.0.0 198.5.203.3 0.0.0.0 UG 1500 0 0 eth0

$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
Tcp 0 127 uranus.toutbet:telnet 194.206.6.143:1027 ESTABLISHED

$ netstat -i
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR
Flags
Lo 3584 0 764 0 0 764 89 0 0 0 BLRU
eth0 1500 0 410856 0 0 33286 210 0 0 0 BRU
```

1. Quels sont les noms et adresse de la machine consultée ?
2. Quel type de session est-elle en train de supporter ?
3. A quoi correspond l'adresse 198.5.203.3?
4. Pourquoi une interface porte-t-elle les Flags BLRU et l'autre BRU ?
5. Quelle est la taille des paquets utilisée par la passerelle par défaut ?

Université Ibn Tofail
Faculté des Sciences
Département d'Informatique
Kénitra

A.U. 2022/2023
Master Génie Logiciel
pour le Cloud Computing
Semestre 1
Machine Learning

Epreuve (2h)

Session Normale

Le jeu de données de travail met en relation 50 attributs descriptifs d'écrivant des caractéristiques architecturales d'immeubles résidentiels (surface des murs, surface des toits, orientation, etc) et 2 attributs cibles : les charges de chauffage (Yheat) et les charges de climatisation (Ycool) de ces immeubles. Le problème d'apprentissage automatique consiste à prédire les attributs cibles, c'est-à-dire les charges, en fonction des attributs descriptifs.

Les données ont été recueillies pour 2000 immeubles et stockées dans un fichier data.csv. Tous les attributs descriptifs sont numériques. Pour les charger en Python, on utilisera le code suivant:

```
import numpy as np
data = np.loadtxt("./data.csv")
X = data[:,:-2]
Y = data[:,-2:]
Yheat = Y[:,0]
Ycool = Y[:,1]
```

1. **Problème de régression :**

Dans un premier temps, on ne va s'intéresser qu'à la variable **Yheat** : la charge de chauffage.

(a) Réservez **25%** des exemples pour le **test** et **75%** pour **d'entrainement**.

(b) Faire standardiser les données.

(c) Apprenez un modèle de régression linéaire **Fheat** permettant de prédire les charges de chauffage en fonction des attributs descriptifs en minimisant la fonction coût RMSE.

(d) Affichez les scores (RMSE et r2_score) de ce modèle de régression calculé sur l'échantillon **test**. Expliquez précisément ce que représentent ces scores.

(e) On peut tenter d'améliorer le score. Pour cela, remplacer l'étape (c) pour procéder par validation croisée.

2. **Problème de classification**

On peut transformer le problème initial en un problème de classification. Par une méthode de clustering, on peut répartir les charges de chauffage et de climatisation en 3 classes : **faibles, moyennes, élevées** codées 0, 1, 2 respectivement.

Les étiquettes des classes se trouvent dans le fichier "**./labels**". Le code Python permettant de charger ce fichier :

```
target = np.loadtxt('./labels', dtype='int')
```

Nous allons comparer plusieurs méthodes d'apprentissage supervisé :

1. les k-plus proches voisins

   model1 = KNeighborsClassifier(n_neighbors= k)

   Hyperparamètre à régler : k.

2. Régression logistique

   Model2= LogisticRegression ()

   Hyperparamètre à régler : aucun

3. SVM à noyau rbf

   Model3 = SVC(gamma= g, kernel = 'rbf')

   Hyperparamètre à régler : g.

4. les arbres de décision

   Model4= tree.DecisionTreeClassifier()

   Hyperparamètre à régler : aucun.

Ecrivez les codes permettant de :

(a) séparer les données de travail en des données d'entrainement et de test,

(b) sélectionner les hyperparamètres des algorithmes d'apprentissage sur l'échantillon d'entrainement par validation croisée,

(c) afficher le score accuracy de ces algorithmes évalués sur l'échantillon test.

3. Clustering :

Par les méthodes de clustering, on peut répartir le jeu de données en plusieurs classes selon les charges de chauffage et de climatisation (**Yheat**, **Ycool**). On va utiliser et comparer les trois méthodes suivantes :

- K-means
- Hiérarchique CHA
- DBSCAN

(a) Décrire en bref les avantages et inconvénients de chaque méthode.

(b) Ecrire le code pour chacune des méthodes citées.

(c) Quels sont les indices ou coefficients à utiliser pour évaluer une méthode de clustering ? Comment peut-on faire pour déterminer le choix optimal des hyperparamètres ?

4. Réduction de la dimension

Notre jeu de données **X = data[:,:-2]** contient 50 attributs descriptifs, on veut réduire la dimension du jeu de données X par l'analyse en composante principale (ACP).

(a) Quel est l'objectif de la réduction de la dimension ?

(b) Appliquer l'ACP à 5 composantes principales afin de réduire la dimension des données X.

(c) Afin de visualiser les résultats de classification de la partie 2. Effectuer les projections des données test sur les deux premiers axes principaux.

(d) Visualiser ces projections des données test étiquetées avec les classes prédites par le modèle 2 de régression logistique.

---------- **ANNEXE** ----------

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn. neighbors import KNeighborsClassifier
knnclass = KNeighborsClassifier(n_neighbors=7)
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn import tree
from sklearn import metrics
from sklearn import cluster
from sklearn.decomposition import PCA

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=25,stratify=y)
scaler = StandardScaler().fit(X_train)
X_train =scaler.transform(X_train)
rmse= metrics.mean_squared_error(y_test, y_pred, squared=False)))
Le coefficient de détermination : R2 = metrics.r2_score(y_test, y_pred)
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
a = metrics.accuracy_score (y_test, y_pred)
kfold = KFold(n_splits=5, random_state=25, shuffle=True)
```

```
results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
knnclass_cv=model_selection.GridSearchCV( model, params, cv=5, scoring='accuracy')

k_values = [5, 7, 9, 13, 15, 17, 19, 21]
params={'n_neighbors': k_values}
knnclass_cv.best_params_['n_neighbors']
knnclass_cv.best_score_
y_pred = knnclass_cv.best_estimator_.predict(X_test)

km = cluster.KMeans(n_clusters=k)
dbscan = cluster.DBSCAN(eps=0.1, min_samples=2)
cha = cluster.AgglomerativeClustering(linkage="ward", distance_threshold = 6, n_clusters=None)
km.fit(X)
cs= metrics.silhouette_score(X, km.labels_)
ir=metrics.adjusted_rand_score(labels, km.labels_)
pca = PCA(n_components=5)
```