Module:	**Machine Learning**
Niveau:	**2ème Année**
Filière:	**CI INFO**
Volume Horaire:	**42 Heures**
Professeur:	**Dr. Ing. Khalid OQAIDI**

Règlement du Cours

- **Discussions entre étudiants** : Les échanges entre élèves sont interdits pendant les interventions du professeur.
- **Questions** : Pour poser une question, levez la main et attendez votre tour.
- **Participation** : La discipline et la participation sont prises en compte dans la note finale de validation.

Modes d'examen

La note finale est moyenne pondérée de:

- **Travail en classe 20 % :** **Quiz à l'improviste + participation et discipline.**
- **Contrôle (30 %) :** **Projet.**
- **Examen final (50%) :** **Ecrit.**

Plan du cours

1. **Introduction à Python pour la Data Science et Machine Learning**
2. **Manipulation des Données avec Pandas et NumPy**
3. **Visualisation des Données avec Matplotlib, Seaborn et Plotly**
4. **Web Scraping et Acquisition de Données**
5. **Prétraitement des Données et Feature Engineering**
6. **Introduction au Machine Learning et Régression Linéaire**
7. **Classification avec Régression Logistique et KNN**
8. **Arbres de Décision et Forêts Aléatoires**
9. **Support Vector Machines (SVM) et Clustering (K-Means)**
10. **Réduction de Dimension avec PCA et Introduction au NLP**
11. **Validation et Optimisation des Modèles**
12. **Introduction aux Réseaux de Neurones avec TensorFlow/Keras**
13. **Déploiement et Intégration des Modèles**
14. **Projet Final en Machine Learning**

Test de Diagnostic et Prérequis

Installation des outils et librairies

Anaconda – jupyter notebook (anaconda.com)

Com*mande pour installer les librairies:*

pip install ……

numpy

pandas

matplotlib

seaborn

plotly

beautifulsoup4

sickit-learn

scipy

nltk

spacy

tenserflow

torch

…

conda install numpy pandas …..

pip install numpy pandas matplotlib seaborn plotly scikit-learn scipy beautifulsoup4 nltk spacy tensorflow torch

Opérations sur les listes

Les listes sont l'une des structures de données les plus couramment utilisées en Python. Elles permettent de stocker plusieurs éléments dans une seule variable et sont modifiables (mutables).

Exemple de création de liste :

```
# Liste de nombres

nombres = [1, 2, 3, 4, 5]

# Liste de chaînes de caractères

fruits = ["pomme", "banane", "cerise"]
```

```
# Liste mixte

mixte = [1, "pomme", True]
```

Opérations sur les listes

Accéder aux Éléments d'une Liste

Exemple d'accès aux éléments d'une liste :

```
# Premier élément
print(fruits[0])  # "pomme"

# Dernier élément
print(fruits[-1])  # "cerise"

# Tranches (slicing)
print(fruits[0:2])  # ['pomme', 'banane']
```

Modifier une Liste

Exemple de modification d'une liste :

```
# Modifier un élément
fruits[1] = "orange"

# Ajouter un élément à la fin
fruits.append("mangue")

# Insérer un élément à un index donné
fruits.insert(1, "kiwi")

print(fruits)  # ["pomme", "kiwi", "orange", "cerise", "mangue"]
```

Opérations sur les listes

Supprimer des Éléments

Exemples de suppression d'éléments d'une liste :

```
# Supprimer un élément par sa valeur
fruits.remove("cerise")

# Supprimer un élément par son index
del fruits[0]

# Supprimer le dernier élément
fruits.pop()

# Supprimer tous les éléments
fruits.clear()

print(fruits)  # []
```

Parcourir une Liste

Exemple de parcours d'une liste avec une boucle :

```
fruits = ["pomme", "banane", "cerise"]

# Parcourir avec une boucle for
for fruit in fruits:
    print(fruit)

# Parcourir avec une boucle for et index
for i in range(len(fruits)):
    print(f"Index {i}: {fruits[i]}")
```

Opérations sur les listes

Rechercher dans une Liste

Exemple de recherche dans une liste :

```
# Vérifier si un élément est présent
print("pomme" in fruits)  # True

# Trouver l'index d'un élément
index = fruits.index("banane")
print(index)  # 1
```

Opérations Mathématiques sur Listes de Nombres

Exemple de calculs avec des listes de nombres :

```
nombres = [10, 20, 30, 40]
# Somme des éléments
print(sum(nombres))  # 100

# Minimum et Maximum
print(min(nombres))  # 10
print(max(nombres))  # 40

# Tri des éléments
nombres.sort()
print(nombres)  # [10, 20, 30, 40]

# Tri inversé
nombres.sort(reverse=True)
print(nombres)  # [40, 30, 20, 10]
```

Opérations sur les listes

Copier une Liste

Exemple de copie d'une liste :

```
fruits = ["pomme", "banane", "cerise"]

# Copier une liste
copie = fruits.copy()

# Vérifier que c'est une copie indépendante
copie.append("kiwi")

print(fruits)  # ["pomme", "banane", "cerise"]
print(copie)   # ["pomme", "banane", "cerise", "kiwi"]
```

Fusionner des Listes

Exemple de fusion de listes :

```
liste1 = [1, 2, 3]
liste2 = [4, 5, 6]

# Concaténer deux listes
liste3 = liste1 + liste2

# Étendre une liste existante
liste1.extend(liste2)

print(liste3)  # [1, 2, 3, 4, 5, 6]
print(liste1)  # [1, 2, 3, 4, 5, 6]
```

Opérations sur les listes

Compréhension de Liste (List Comprehension)

Exemple de compréhension de liste :

```
# Créer une liste des carrés des nombres de 0 à 4
carrés = [x**2 for x in range(5)]
print(carrés)  # [0, 1, 4, 9, 16]

# Filtrer une liste (garder uniquement les nombres pairs)
pairs = [x for x in range(10) if x % 2 == 0]
print(pairs)  # [0, 2, 4, 6, 8]
```

Exercices (listes)

Exercice: Transformation et Analyse d'une Liste Numérique

1. Générez une liste `nombres` contenant les 30 premiers nombres entiers positifs.
2. Filtrez uniquement les nombres multiples de 3.
3. Élevez chaque élément filtré au carré et stockez-les dans une nouvelle liste `carres`.
4. Trouvez et affichez la somme des éléments pairs de `carres`.
5. Affichez la liste `carres` triée en ordre décroissant.

Exercice: Manipulation de Textes sous Forme de Liste

1. Créez une liste `phrases` contenant au moins 6 phrases de votre choix.
2. Trouvez la phrase la plus longue et affichez-la.
3. Trouvez le nombre total de mots dans toutes les phrases combinées.
4. Créez une nouvelle liste `voyelles` contenant uniquement les phrases qui commencent par une voyelle (a, e, i, o, u).
5. Affichez les phrases de `voyelles`, chacune en majuscule.

Opérations sur les dictionnaires

Les dictionnaires sont des structures de données en Python qui stockent des paires clé-valeur. Ils permettent un accès rapide aux données et sont largement utilisés en Data Science et Machine Learning.

Création d'un Dictionnaire

Exemple de création d'un dictionnaire :

```
# Dictionnaire simple
etudiant = {
   "nom": "Ali",
   "âge": 21,
   "matières": ["Maths", "Physique", "Informatique"]
}

# Dictionnaire vide
dico_vide = {}
```

Accéder aux Éléments d'un Dictionnaire

Exemple d'accès aux éléments d'un dictionnaire :

```
# Accéder à une valeur via sa clé
print(etudiant["nom"])  # "Ali"

# Utiliser .get() pour éviter une erreur si la clé n'existe pas
print(etudiant.get("âge", "Clé non trouvée"))  # 21

# Accéder à un élément dans une liste stockée dans le dictionnaire
print(etudiant["matières"][1])  # "Physique"
```

Opérations sur les dictionnaires

Modifier un Dictionnaire

Exemple de modification d'un dictionnaire :

```
# Modifier une valeur existante
etudiant["âge"] = 22

# Ajouter une nouvelle clé
etudiant["moyenne"] = 16.5

# Afficher le dictionnaire mis à jour
print(etudiant)
```

Supprimer des Éléments d'un Dictionnaire

Exemples de suppression d'éléments d'un dictionnaire :

```
# Supprimer une clé spécifique
del etudiant["moyenne"]

# Supprimer une clé avec .pop()
age = etudiant.pop("âge")

# Vider complètement un dictionnaire
etudiant.clear()

print(etudiant)  # {}
```

Opérations sur les dictionnaires

Parcourir un Dictionnaire

Exemple de parcours d'un dictionnaire avec une boucle :

```
# Parcourir les clés
for clé in etudiant.keys():
    print(clé)

# Parcourir les valeurs
for valeur in etudiant.values():
    print(valeur)

# Parcourir les paires clé-valeur
for clé, valeur in etudiant.items():
    print(f"{clé} : {valeur}")
```

Vérifier l'Existence d'une Clé

Exemple de vérification de la présence d'une clé dans un dictionnaire :

```
if "nom" in etudiant:
    print("La clé 'nom' existe dans le dictionnaire.")
```

Opérations sur les dictionnaires

Fusionner Deux Dictionnaires

Exemple de fusion de dictionnaires :

```
infos1 = {"nom": "Sara", "âge": 25}
infos2 = {"ville": "Casablanca", "profession": "Ingénieur"}

# Fusionner avec update()
infos1.update(infos2)

print(infos1)
```

Générer un Dictionnaire avec une Compréhension

Exemple de génération d'un dictionnaire en utilisant une compréhension :

```
# Générer un dictionnaire de carrés des nombres de 1 à 5
carres = {x: x**2 for x in range(1, 6)}

print(carres)  # {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Opérations sur les dictionnaires

Trier un Dictionnaire

Exemple de tri d'un dictionnaire par ses valeurs :

```
scores = {"Ali": 85, "Sara": 92, "Omar": 78}

# Trier par score (ordre croissant)
scores_tries = dict(sorted(scores.items(), key=lambda x: x[1]))

print(scores_tries)  # {'Omar': 78, 'Ali': 85, 'Sara': 92}
```

Opérations sur les dictionnaires

Compter les Occurrences dans une Liste avec un Dictionnaire

Exemple de comptage des occurrences d'éléments dans une liste :

```
mots = ["chat", "chien", "chat", "oiseau", "chien", "chien"]

# Créer un dictionnaire de comptage
frequence = {}

for mot in mots:
    frequence[mot] = frequence.get(mot, 0) + 1

print(frequence)  # {'chat': 2, 'chien': 3, 'oiseau': 1}
```

Exercices (listes)

Exercice: Compression de Liste et Optimisation

1. Générez une liste `grande_liste` contenant 100 nombres entiers aléatoires entre 1 et 1000.
2. Supprimez les doublons tout en conservant l'ordre initial.
3. Créez un dictionnaire de fréquences indiquant combien de fois chaque nombre apparaît dans la liste d'origine.
4. Affichez les 5 nombres les plus fréquents et leurs occurrences.
5. Réduisez la liste en ne gardant que les nombres apparaissant plus de 2 fois.

Exercices (dictionnaires)

Exercice: Gestion d'un Inventaire

1. Créez une liste `inventaire` contenant au moins 5 dictionnaires, chacun représentant un produit avec :
   - `"nom"` (str),
   - `"quantité"` (int),
   - `"prix_unitaire"` (float).
2. Trouvez et affichez le produit le plus cher.
3. Trouvez et affichez le produit ayant le plus grand stock.
4. Calculez le coût total du stock (quantité × prix).
5. Supprimez du stock tous les produits avec une quantité inférieure à 5.

Exercice 1 : Analyse des Notes des Étudiants

1. Créez un dictionnaire `notes_etudiants` où les clés sont les noms des étudiants et les valeurs sont leurs moyennes.
2. Trouvez et affichez l'étudiant ayant la meilleure note.
3. Affichez le moyen général de la classe.
4. Créez une nouvelle liste des étudiants ayant une note ≥ 12.
5. Triez les étudiants par ordre décroissant de note et affichez le dictionnaire trié.

Exercice 2 : Gestion d'un Stock de Produits

1. Créez une liste `inventaire` contenant 5 dictionnaires, représentant chacun un produit avec :
   - `"nom"` (str)
   - `"quantité"` (int)
   - `"prix_unitaire"` (float)
2. Trouvez et affichez le produit le plus cher.
3. Trouvez et affichez le produit ayant le plus grand stock.
4. Calculez le coût total du stock (`quantité × prix_unitaire`).
5. Supprimez du stock les produits avec une quantité < 5.

Exercice 3 : Analyse de Fréquence des Mots

1. Demandez à l'utilisateur d'entrer une phrase.
2. Transformez cette phrase en liste de mots.
3. Créez un dictionnaire de fréquence des mots.
4. Affichez les 3 mots les plus fréquents.

Exercice 4 : Convertir un Dictionnaire en Liste

1. Définissez un dictionnaire `capitale_pays` contenant 5 pays et leurs capitales.
2. Convertissez ce dictionnaire en une liste de tuples.
3. Convertissez la liste obtenue en un nouveau dictionnaire.

Dataframe & Series