TP	Traçabilité 4.0 Etude de la traçabilité dans la production de lait	Période				
		1	2	3	4	5
Cycle 3 : TRACABILITE	Durée : 3 semaines					X

1- Les différentes étapes de la traçabilité 4.0

Objectif

Assurer une traçabilité complète et transparente du lait depuis la ferme jusqu'au consommateur final, en utilisant des données en temps réel collectées par des capteurs pour améliorer la qualité, la sécurité et l'efficacité opérationnelle.

Technologies Utilisées

1. **Internet des Objets (IoT)** :
   - Capteurs de santé des vaches (température corporelle, activité, production laitière).
   - Capteurs de température et d'humidité dans les camions de transport.
   - Capteurs de qualité du lait (pH, bactériologie) dans l'usine de transformation.
2. **Blockchain** :
   - Enregistrement immuable des données de traçabilité.
3. **Big Data et Analytics** :
   - Collecte et analyse des données pour optimisation et prévention.
4. **Cloud Computing** :
   - Stockage des données pour un accès partagé.
5. **Intelligence Artificielle (IA) et Machine Learning (ML)** :
   - Prédiction et optimisation des processus.

2- Mise en œuvre du process

1. **Ferme** :
   - Installation de capteurs IoT sur 100 vaches pour surveiller la température corporelle, l'activité et la production laitière.
   - Les données sont collectées toutes les heures et envoyées à une plateforme cloud.
2. **Transport** :
   - Équipement de 10 camions avec des capteurs de température et d'humidité.
   - Les données sont collectées en continu et transmises en temps réel via le cloud.
3. **Usine de Transformation** :
   - Capteurs de pH et de niveau de bactéries installés dans les cuves de réception du lait.
   - Analyse des données en temps réel pour détecter toute anomalie.
4. **Blockchain** :
   - Chaque lot de lait est enregistré sur une blockchain, incluant les données de santé des vaches, les conditions de transport et les analyses de qualité en usine.
   - Les consommateurs peuvent accéder aux informations via un QR code sur l'emballage.

1) Ouvrir avec le logiciel Spyder, le fichier lait.py. Dans ce script est présent le code qu'il faudra au fur et mesure de l'activité.

On souhaite générer des données aléatoires issues des différents capteurs cités précédemment. Pour ce faire, nous allons créer les variables suivantes :
num_vaches : qui représente le nombre de vaches : 100
num_jours : le nombre de jours pour lesquels nous allons générer les données : 30
temps_de_transport : le nombre d'heures de transport : 3
heures_par_jour : le nombre d'heures de surveillance par jour : 24

2) Rajouter ces variables globales dans le script python.

3) Créer une variable nommée « dates » dans laquelle nous allons générer 30 dates pour chaque jour du mois de janvier 2024 avec l'instruction suivante :
pd.date_range(start='aaaa-mm-dd', periods=nbr de dates, freq='D')

La génération de données aléatoires pour les différents est renseignée dans le code fourni, elles sont stockées dans les variables suivantes :

Données des Capteurs pour les Vaches :
- température corporelle (temperature_vaches) en degré,
- activité (activite_vaches) en heures,
- production laitière des vaches (production_lait) en litres.

Données de Transport :
- température (temperature_transport) en degré,
- humidité dans les camions de transport (humidite_transport) en pourcentage.

Données de l'Usine :
- pH du lait (ph_lait) sans unité
- niveau de bactéries en usine (niveau_bacteries) compris entre 0 et 1000.

En vue d'assurer la traçabilité, il est nécessaire par la suite de renseigner ces valeurs dans des bases de données accessibles par tous les acteurs. Ces bases de données sont appelées dataframe en langage python.

Les bases de données du transport et de la santé des vaches sont déjà implémentées dans le code.

4) En vous aidant du code existant, compléter le script pour créer une base de données nommée df_usine dans laquelle devra apparaître la date, le pH du lait ainsi que le niveau de bactérie pour les 30 jours. La base devra ressembler à l'exemple ci-dessous :

Indice	Date	pH_Lait	Niveau_Bactéries
0	2024-01-01	6.8387	883.338
1	…	…	…

5) Afficher, pour vérification du contenu des bases de données, les premières lignes de chaque dataframe en utilisant l'instruction : print(nom_dataframe.head()). Cela devrait afficher les 5 premiers enregistrements de la base de données.

6) Décommenter la partie suivante du code pour afficher le graphique de température des vaches, du transport ainsi que du niveau de bactéries.

7) Créer une fonction appelée « moyenne » qui permet de déterminer la valeur moyenne des valeurs qui composent un tableau de « l » lignes et « c » colonnes. Tester la fonction avec la variable temperature_vaches.

8) Créer une fonction nommée « variabilite » qui affiche par rapport à la moyenne issue d'un tableau, l'écart maximum en positif et minimum en négatif. Tester cette fonction avec la température des vaches.

3- Maintenance Prédictive par Machine Learning

Pour effectuer de la maintenance prédictive à partir des données générées, vous pouvez utiliser plusieurs techniques d'analyse de données, principalement basées sur l'apprentissage automatique (Machine Learning). L'objectif est de prédire quand un équipement ou une vache, dans ce cas, pourrait avoir besoin d'intervention pour prévenir des pannes ou des baisses de performance.

Étapes pour la Maintenance Prédictive

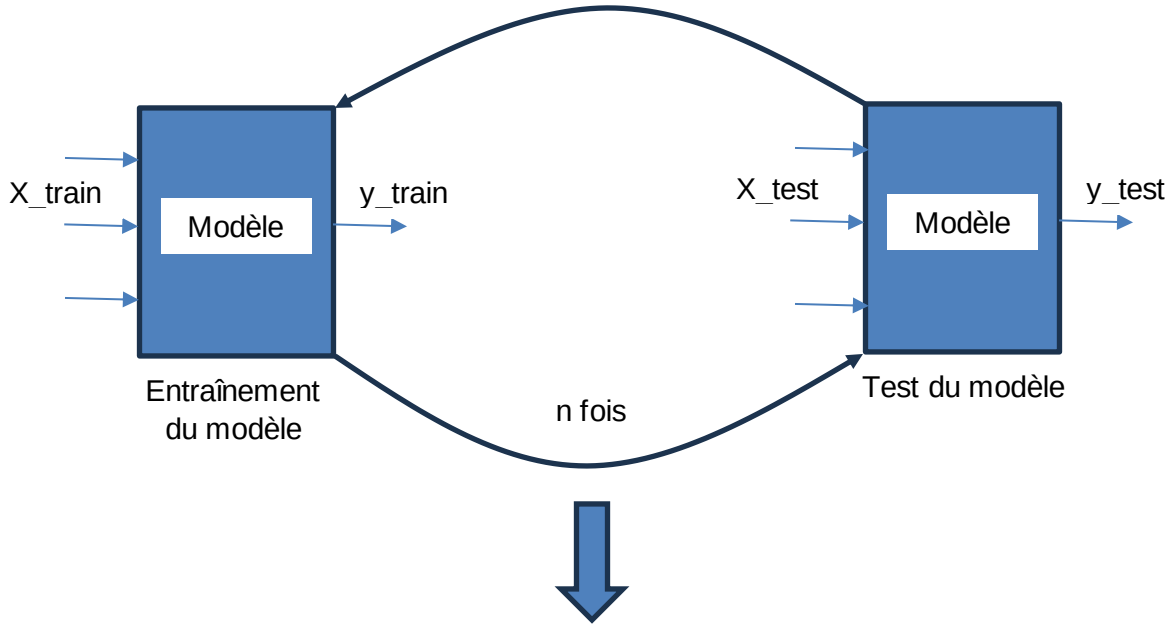
1. **Exploration des Données** :
   - Analyser les tendances et identifier les comportements anormaux dans les données.
   - Par exemple, observer si certaines vaches montrent une baisse de production de lait ou une température corporelle anormale avant de développer des problèmes de santé.
2. **Préparation des Données** :
   - **Nettoyage** : Remplir les valeurs manquantes ou éliminer les données erronées.
   - **Feature Engineering** : Créer de nouvelles variables basées sur les données existantes. Par exemple, vous pourriez créer une variable qui représente le taux de changement de la température corporelle d'une vache au fil du temps.
   - **Sélection des Caractéristiques** : Choisir les caractéristiques les plus pertinentes (par exemple, température, activité, production laitière).
3. **Modélisation avec le Machine Learning** :
   - Choisir un algorithme de Machine Learning pour prédire les événements futurs. Les algorithmes couramment utilisés incluent :
     - **Régression Logistique** : Pour des prédictions binaires (p. ex., prédire si une vache tombera malade ou non).
     - **Arbres de Décision ou Forêts Aléatoires** : Pour identifier des relations complexes entre les caractéristiques.
     - **Réseaux de Neurones** : Pour des données complexes ou volumineuses.
4. **Entraînement du Modèle** :
   - **Séparation des Données** : Diviser les données en ensembles d'entraînement et de test.
   - **Entraînement** : Former le modèle sur l'ensemble d'entraînement.
   - **Validation Croisée** : Évaluer le modèle pour s'assurer qu'il généralise bien aux nouvelles données.
5. **Évaluation du Modèle** :
   - Utiliser des métriques comme la précision, le rappel, et le score F1 pour évaluer la performance du modèle.
   - Par exemple, si vous prédisez la maladie d'une vache, vérifier combien de fois le modèle a correctement identifié les cas malades par rapport aux faux positifs et faux négatifs.
6. **Interprétation et Décision** :
   - Identifier les facteurs clés qui contribuent à la probabilité de panne ou de maladie.
   - Générer des alertes basées sur les prédictions (exemple, si une vache a une probabilité élevée de tomber malade dans les prochains jours, recommander une intervention vétérinaire).
7. **Déploiement et Surveillance** :
   - Déployer le modèle dans l'environnement opérationnel pour surveiller les nouvelles données en temps réel.
   - Mettre à jour et réentraîner le modèle régulièrement avec de nouvelles données pour améliorer ses performances.

Sélection des Caractéristiques : Nous sélectionnons la température, l'activité et la production laitière comme caractéristiques pour prédire si une vache est en mauvaise santé.

9) Créer une base de données nommée « features » qui prend comme attribut : 'Temperature', 'Activite' et 'Production_Lait'.

10) A partir de ces caractéristiques, nous allons définir une variable nommée « target » qui prendra des valeurs True si la température de la vache est supérieure à 39°C ou si la production de lait est inférieure à 20 litres par jour. Sinon, la valeur sera à False.

Modèle RandomForest : Utilisé pour prédire si une vache risque de tomber malade.
L'idée du Machine Learning est d'utiliser une partie des données pour l'entraînement du modèle et une autre partie pour vérification de ce modèle. Plus le volume de données fournies sera important, plus le modèle sera fiable.



Validation du modèle par évaluation :
Matrice de confusion

Ici, les données d'entrées, X_train et X_test seront les attributs présents dans « features », à savoir la température, l'activité et la production de lait.
La sortie y_train et y_test représentera la suspicion de maladie d'une vache (True ou False)
L'instruction pour la division des données par la bibliothèque sklearn est la suivante :
X_train, X_test, y_train, y_test = train_test_split(tableau-entree, sortie, test_size=0.3(% des données utilisées pour le test), random_state=10 (nombre de cycles)).

11) En utilisant la bibliothèque sklearn, réaliser la division des données en ensembles d'entraînement et de test de manière suivante :

- 70% de données pour l'entraînement,
- 30% des données pour le test du modèle,
- Réaliser 42 cycles aléatoires d'entraînement,
- Les variables d'entrée/sorties sont définis sur le schéma ci-dessus.

Une fois le jeu de données séparées, il faut entraîner le modèle, c'est ce qui est réalisé par l'instruction model.fit issue de l'instance RandomForestClassifier. Le modèle est stocké dans la variable nommée « model ».
On peut maintenant réaliser des prédictions avec le modèle établi avec l'instruction suivante :
prediction = model.predict(donnée d'entrée de test)

12) Réaliser la prédiction de suspicion de maladie pour le jeu de test précédemment défini et stocker le résultat dans la variable nommée « y_pred ».

Pour évaluer la véracité du modèle, nous allons afficher le résultat de notre prédiction en comparaison du résultat réel. Pour ce faire, nous allons créer 2 bases de données nommées « reel » et « prediction » pour lesquelles nous allons afficher le résultat de suspicion de maladie en fonction de la température de la vache, de son activité ainsi que de la production de lait.
L'instruction qui permet de réaliser ces différents graphiques est la suivante :
sns.pairplot(base_de_donnée, hue='critère_évalué')

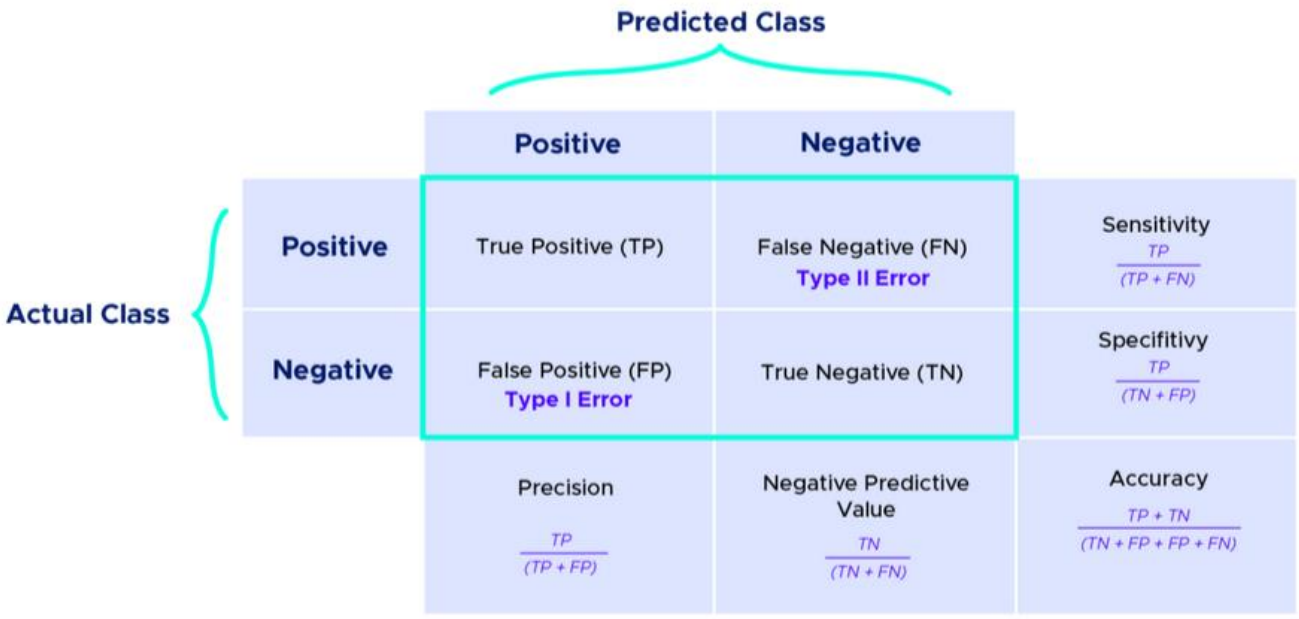
13) Créer la base de données nommée « reel » à partir des bases « df_vaches » et « target » ayant pour attributs : 'Temperature', 'Production_lait', 'Activite', 'malade'.

14) Afficher les graphiques qui permettent de détecter les vaches susceptibles d'être malades.

15) Créer la base de données nommée « prediction » à partir des bases « X_test » et « y_pred » ayant pour attributs : 'Temperature_p', 'Production_lait_p', 'Activite_p', 'malade_p'.

16) Afficher les graphiques qui permettent de détecter les vaches susceptibles d'être malades.

En comparant les deux séries de graphiques, nous pouvons constater les différences éventuelles entre la réalité et la prédiction. Cependant, il est difficile de quantifier le taux d'erreur de notre prédiction.
La matrice de confusion est l'outil utilisé pour évaluer la qualité du modèle que nous avons entraîné.



Il permet de quantifier la précision du modèle entraîné comme expliqué sur le schéma ci-dessus.

17) Réaliser la matrice de confusion en décommentant la dernière partie du code et conclurte sur la validité du modèle à partir du rapport de classification.