Projet : Système Intelligent de Sélection de Modèles via Meta-Learning

Détection d'Occupation de Salles par Capteurs IoT (Version 6 Features)

1. Objectif Général

Concevoir une architecture de meta-learning capable de sélectionner dynamiquement le modèle de classification optimal pour détecter l'occupation d'une salle à partir de mesures environnementales. L'objectif pédagogique est de maîtriser les concepts d'ensemble learning, de meta-features, et d'architecture multi-niveaux appliqués à un problème réel de gestion énergétique.

2. Contexte

La détection automatique d'occupation permet d'optimiser la consommation énergétique des bâtiments intelligents (éclairage, climatisation, chauffage) avec des économies de 20 à 40 %. Le dataset d'occupation de salles contient environ 20 000 mesures de capteurs (température, humidité, lumière, CO_2, humidity ratio, index temporel) avec labels d'occupation.

Question centrale : Face à des données temporelles complexes et corrélées, comment sélectionner intelligemment le meilleur algorithme pour chaque nouvelle mesure ?

3. Concept du Meta-Learning

L'approche repose sur deux niveaux d'apprentissage :

3.1 Niveau 1 - Classifieurs de Base

Entraînement de quatre modèles distincts :
— Decision Tree
— Random Forest
— Support Vector Machine (SVM)
— Naive Bayes

Chaque modèle capture différemment les patterns d'occupation selon sa nature mathématique. Aucun n'est universellement optimal.

3.2 Niveau 2 - Meta-Modèle KNN

Un KNN sélectionne quel classifieur utiliser selon les caractéristiques de la mesure.

Principe durant l'entraînement :

— Pour chaque mesure de validation, passer par les quatre classifieurs
— Identifier quel modèle a donné la bonne réponse
— Enregistrer : les features de la mesure → l'indice du meilleur modèle

Principe durant l'inférence :

— Utiliser KNN pour trouver les $k = 5$ mesures les plus similaires
— Voter : le modèle optimal pour le plus de voisins est sélectionné
— Exécuter ce modèle et retourner sa prédiction

Analogie : Créer une base de références où chaque exemple indique "Decision Tree a réussi ici", "Random Forest là", etc. Pour une nouvelle mesure, on trouve les cas similaires et on utilise le modèle qui y réussissait.

3.3 Avantages

— Ensemble Learning : Combiner plusieurs modèles différents
— Sélection Dynamique : Adapter le modèle aux caractéristiques spécifiques
— Similarité : Exploiter que des problèmes similaires ont des solutions similaires
— Transparence : Comprendre pourquoi un modèle est sélectionné

4. Données

Dataset : Occupancy Detection (version simplifiée inspirée de l'UCI)
Environ 20 000 mesures
6 features :
— Temperature
— Humidity
— Light
— CO_2
— Humidity Ratio
— Time_Index
Label binaire :
— 0 = inoccupé
— 1 = occupé
Équilibré : 55 % inoccupation, 45 % occupation
Lien du jeu de données : `https://archive.ics.uci.edu/dataset/357/occupancy+detection`

5. Architecture Système

Couche 1 : Préparation
— Chargement et normalisation (StandardScaler)
Couche 2 : Classifieurs de Base
— Entraînement indépendant des quatre modèles
— Optimisation des hyperparamètres
— Évaluation individuelle
Couche 3 : Meta-Learning
— Extraction des métafeatures des classifieurs
— Entraînement du KNN (k=5)

— Pipeline de sélection et d'inférence

Couche 4 : Application Web

— Interface Flask pour prédictions interactives

— Traçabilité du processus de sélection

6. Métafeatures

Pour simplifier, on utilise deux métafeatures par classifieur :

1. Confiance Maximale

$$\text{Confiance\_Max} = \max(P(\text{occupé}), P(\text{inoccupé}))$$

2. Margin

$$\text{Margin} = |P(\text{occupé}) - P(\text{inoccupé})|$$

Vecteur final : 4 modèles $\times$ 2 métafeatures = 8 features pour le meta-modèle.

7. Processus d'Entraînement du Meta-Modèle

— Chaque mesure de validation passe par les quatre classifieurs

— Extraction des 8 métafeatures

— Comparaison avec le vrai label

— Le classifieur correct devient la cible

— Fallback sur le plus confiant si échec total

8. Inférence

— Normalisation des features

— Passage par les quatre classifieurs

— Extraction des 8 métafeatures

— KNN identifie les $k = 5$ voisins similaires et vote

— Exécution du classifieur sélectionné

9. Application Flask

Fonctionnalités :

— Formulaire de saisie des features

— Validation des données

— Pipeline complet de prédiction

— Affichage des résultats :

  — Prédiction finale (occupé/inoccupé)

  — Score de confiance

  — Modèle sélectionné

10. Métriques d'Évaluation

— Accuracy
— Precision
— Recall
— F1-Score
— Confusion Matrix

11. Livrables

1. Notebook Jupyter Un notebook unique et complet contenant :
— Chargement et préparation des données
— Entraînement des quatre classifieurs de base
— Extraction des métafeatures
— Entraînement du meta-modèle KNN
— Évaluation et analyses des performances
2. Vidéo Démonstration (5–8 minutes) Vidéo montrant :
— Lancement de l'application Flask
— Saisie de plusieurs exemples de mesures dans le formulaire
— Affichage des prédictions avec :
  — Le modèle sélectionné par le meta-modèle
  — La prédiction finale (occupé/inoccupé)

12. Modalités de Soumission

Date limite : 8 Décembre 2025 à 23h59
Email de soumission : mldiakhame@ugb.edu.sn
Objet du mail : **Projet M2 GDIL DATAMINING - [Code Étudiant 1] ; [Code Étudiant 2]**
Format de soumission : Un fichier ZIP contenant le notebook (.ipynb) et la vidéo démo (.mp4)
Nom du fichier : `Projet_MetaLearning_[Code1]_[Code2].zip`