

AI DETECTION OF HEAVY METALS IN AGRICULTURAL WATER

Réalisé par : Mossab Arektout

Encadré par : Pr.MOUMANE Karima

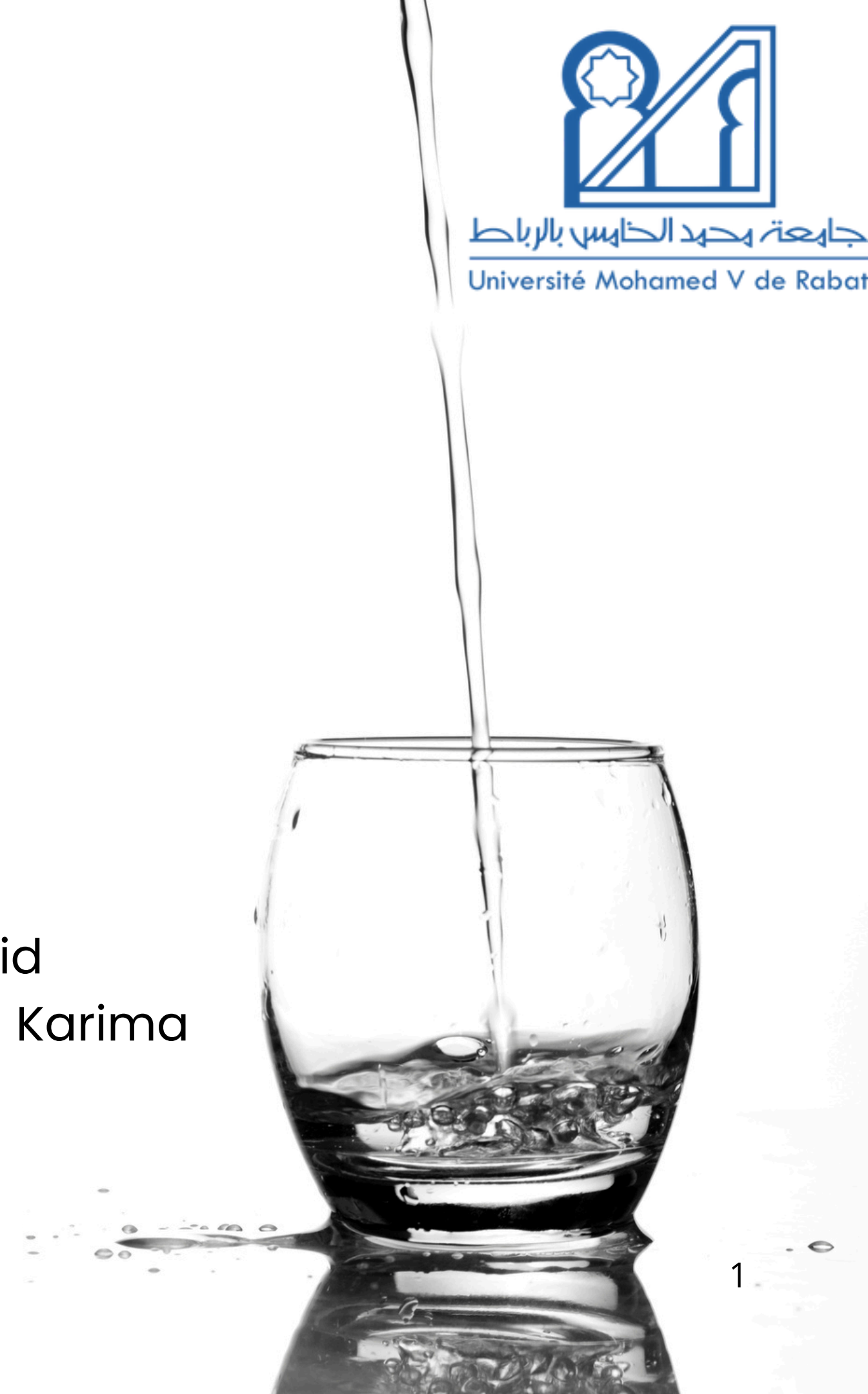
Soutenu devant :

Pr. NAFIL Khalid

Pr. MOUMANE Karima

Data Science and IoT (IDSIT)

Projet S5



Plan de la Présentation

- ➊ Contexte et Problématique
- ➋ Méthodologie et Données
- ➌ Architecture Hiérarchique
- ➍ Résultats Expérimentaux
- ➎ Discussion et Perspectives
- ➏ Conclusion



Contexte et Problematique

Contexte : Une Menace Silencieuse

Les métaux lourds dans l'eau agricole

- **70%** de l'eau douce mondiale utilisée en agriculture
- Contamination par Chrome (Cr) et Arsenic (As)
- **Bioaccumulation** dans la chaîne alimentaire
- Risques sanitaires majeurs :
 - Cancers (peau, poumons, vessie)
 - Troubles neurologiques
 - Insuffisance rénale



La sécurité alimentaire et la santé
publique dépendent d'une
détection rapide et fiable

Problématique : Limites des Méthodes Traditionnelles

Méthodes analytiques actuelles

- Spectrométrie d'absorption atomique (SAA)
- ICP-MS / ICP-OES
- Voltampérométrie

Limitations

- 🏢 Infrastructure lourde requise
- 🎓 Personnel qualifié
- 📍 Inaccessible en zones rurales

Notre solution : IA + SERS

SERS



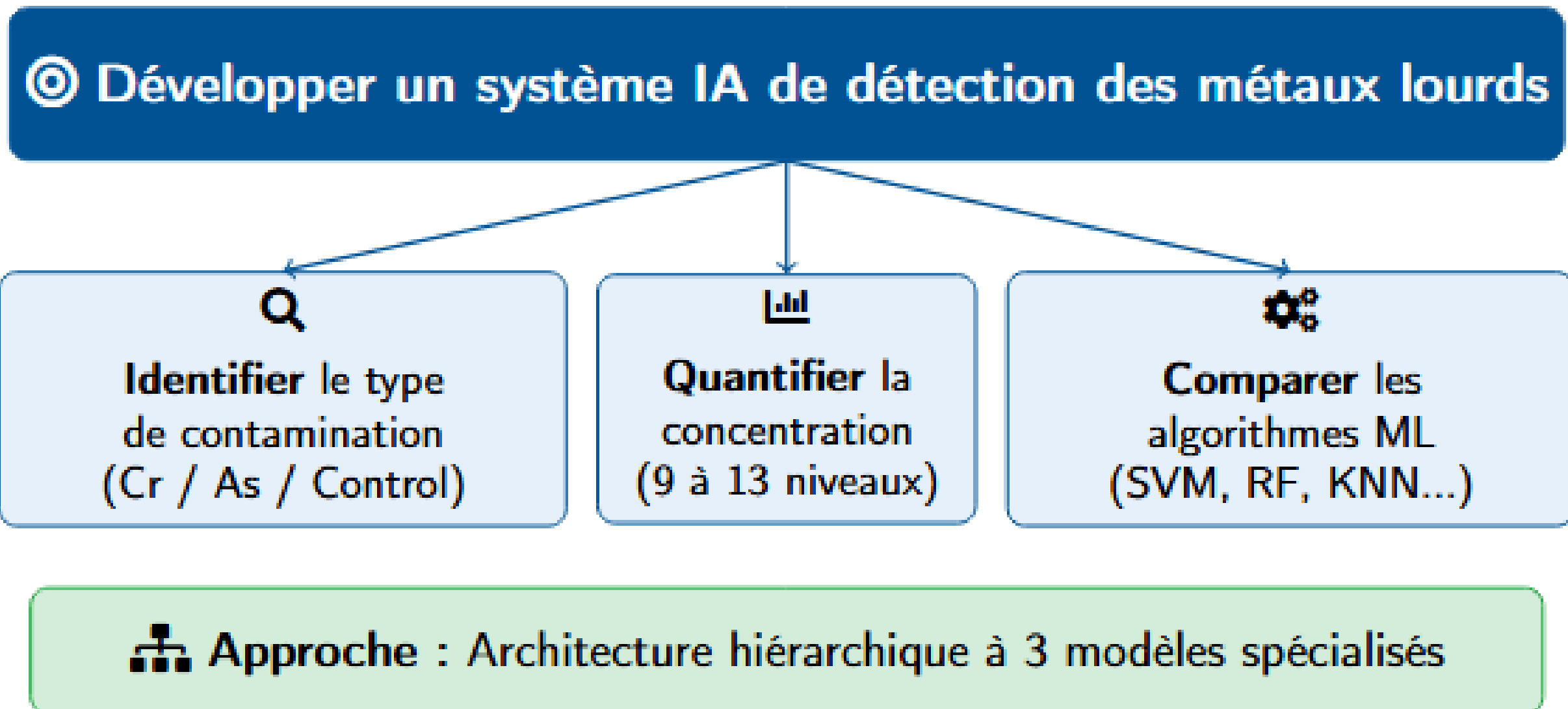
Machine Learning



Détection Rapide

- ✓ Rapide et économique
- ✓ Portable et accessible
- ✓ Haute précision

Objectifs du Projet





Méthodologie et Données

Source des Données : Spectroscopie SERS

Dataset (Wei et al., 2023)

- **Technique** : SERS (Surface-Enhanced Raman Spectroscopy)
- **Biosenseur** : *E. coli*
- **Total** : 42 400 spectres
- **Points/spectre** : 1 011
- **Plage** : 508 - 1640 cm^{-1}

Métal	Spectres	Classes
Chrome (Cr)	18 800	10
Arsenic (As)	23 600	14
Total	42 400	—

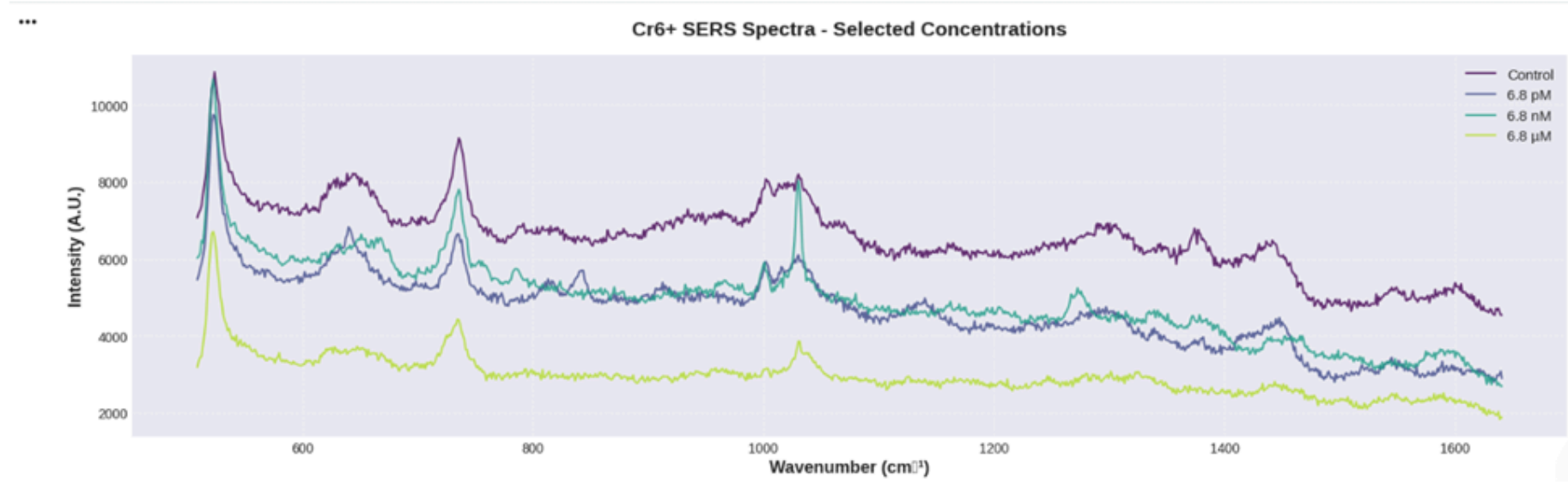


FIGURE 2.3 – Chrome hexavalent (Cr^{6+}) : Control, 68 pM, 68 nM et 68 μM .

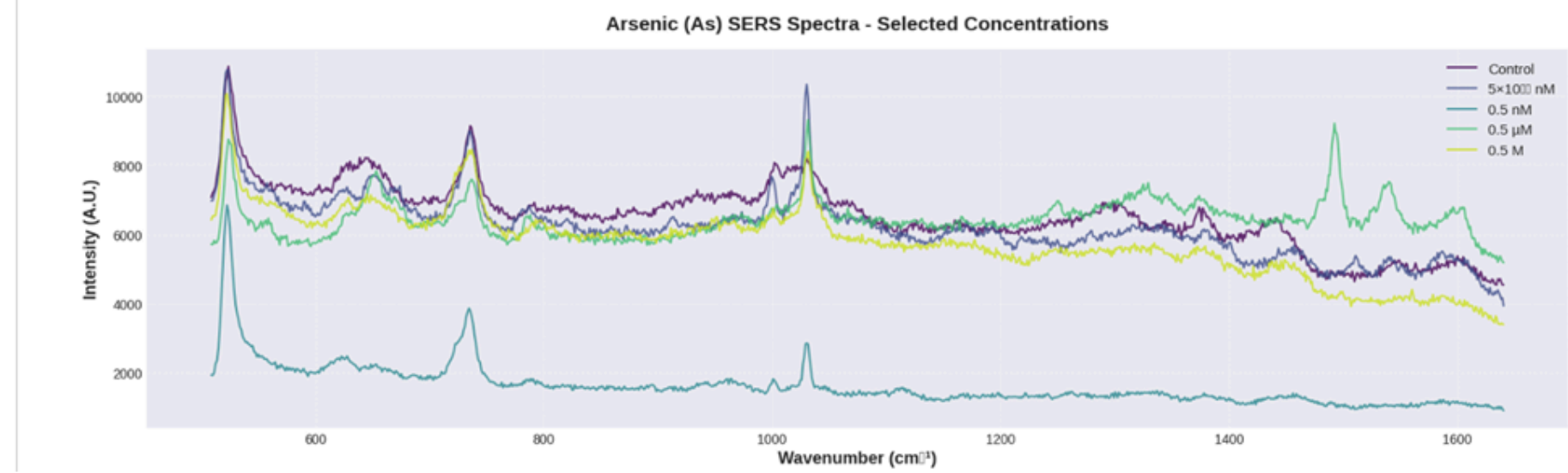
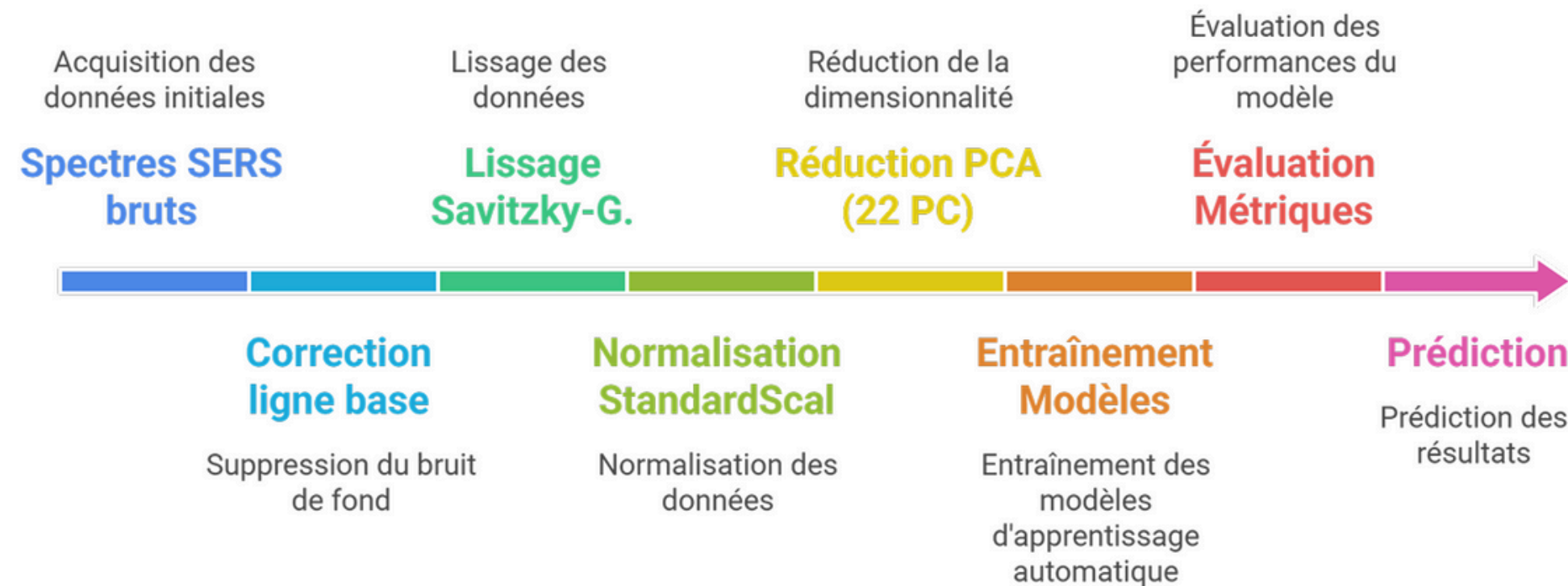


FIGURE 2.4 – Arsenic (As) : Control, 0.05 nM, 50 nM et 50 μM .

Pipeline de Traitement des Données



Division des données

- **80%** Entraînement
- **20%** Test
- Division stratifiée

Équilibrage : SMOTE

- Génération d'échantillons synthétiques
- Appliqué **uniquement** sur le train
- Évite le **bias** vers classes majoritaires



Architecture Hiérarchique

Pourquoi une Architecture Hiérarchique ?

Approche directe (22 classes)

- Classification unique en 22 classes
- Complexité très élevée
- Confusion entre métaux et concentrations
- ✗ Performances limitées

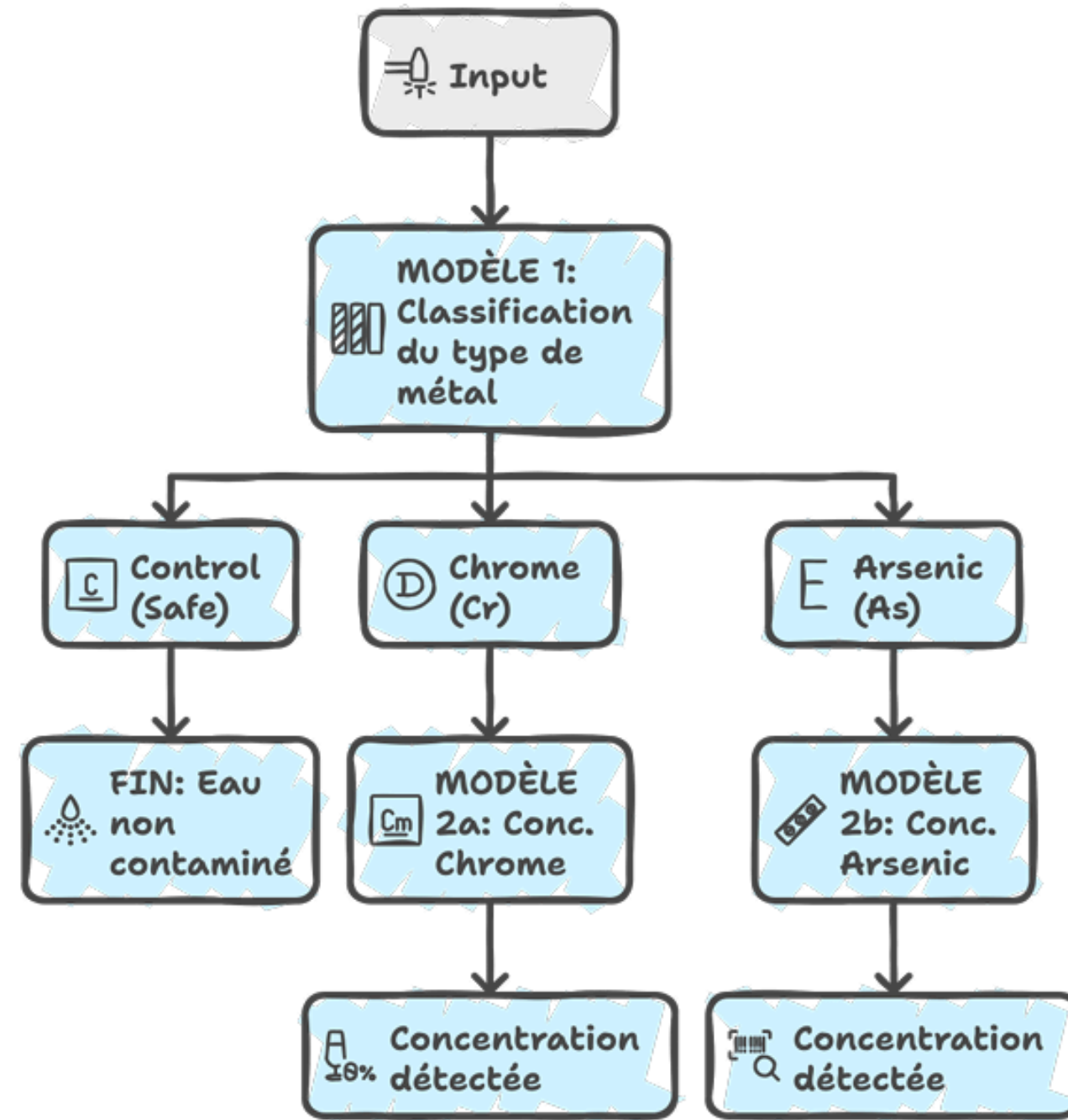
Notre approche hiérarchique

- Décomposition en sous-problèmes
- Modèles spécialisés
- Décision séquentielle logique
- ✓ Meilleures performances

💡 Principe : D'abord identifier QUOI, puis quantifier COMBIEN

Architecture à 3 Modèles

Processus de Détection de Métaux Lourds



Made with Napkin

Algorithmes de Classification Évalués

Algorithme	Principe	Avantages
SVM (RBF)	Hyperplan optimal dans espace de haute dimension	Excellent sur données spectrales
Random Forest	Ensemble d'arbres de décision (vote majoritaire)	Robuste, peu de prétraitement
KNN	Classification par k plus proches voisins	Simple, efficace si classes séparées
Gradient Boosting	Construction séquentielle corrigeant les erreurs	Très bonnes performances

Métriques d'évaluation

Accuracy | Precision | Recall | F1-Score | Matrice de confusion

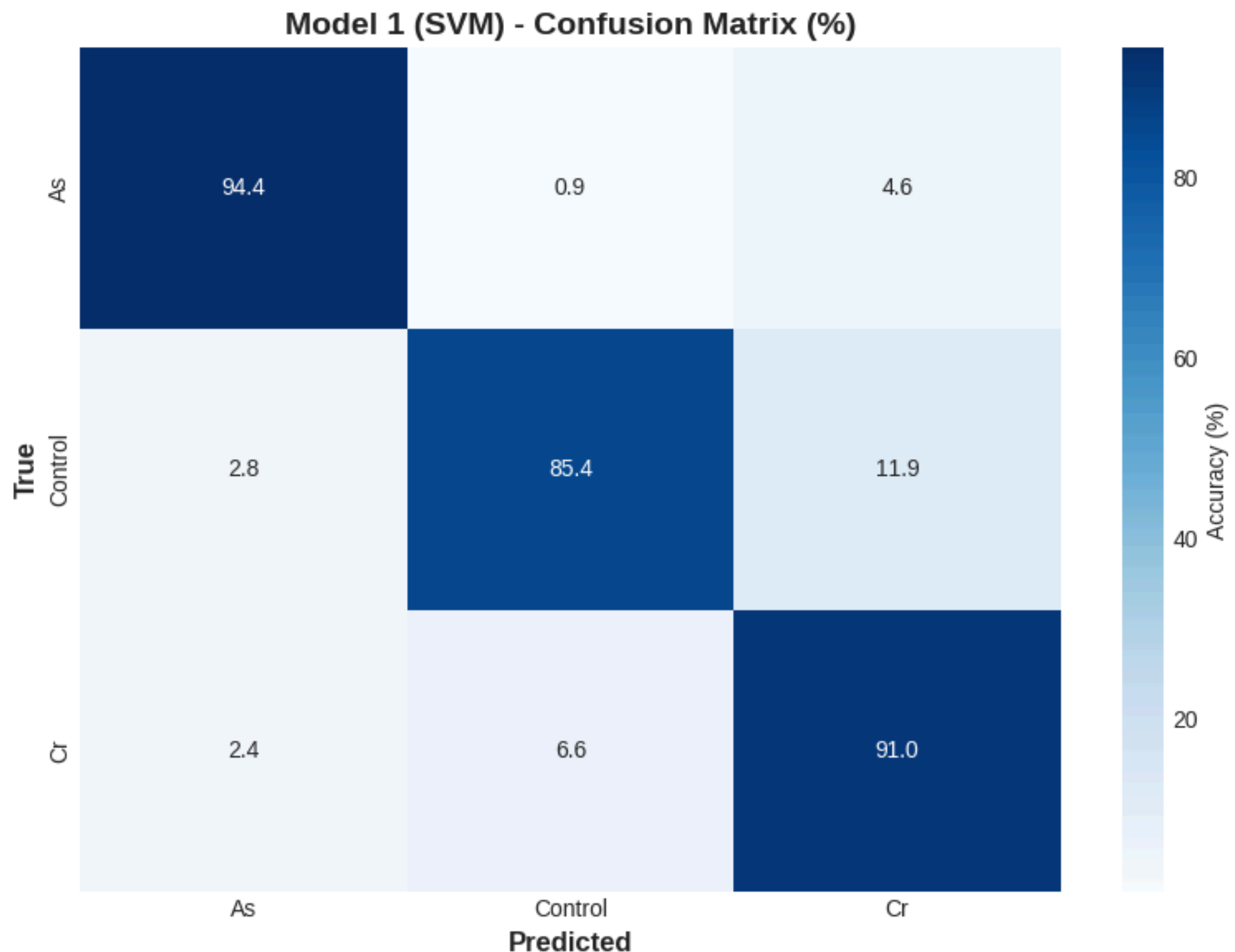


Résultats Expérimentaux

Résultats du Modèle 1 : Classification du Type

Comparaison des algorithmes		
Algo.	Accuracy	F1
SVM	92.01%	0.92
Random Forest	91.71%	0.92

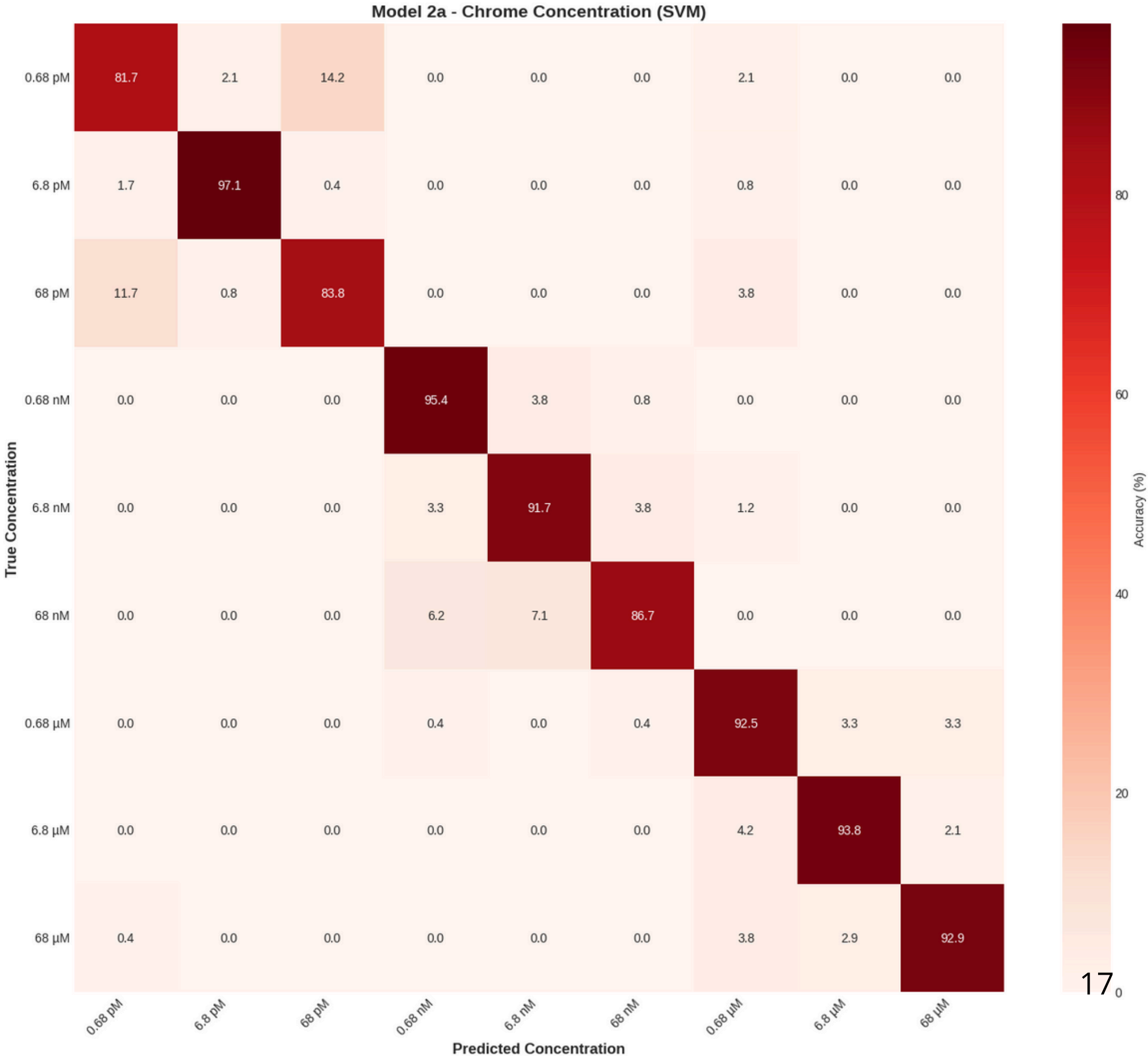
- Points clés
- ✓ Excellente séparation Control/Contaminé
 - ✓ Bonne distinction Cr/As
 - ✓ SVM légèrement supérieur



Résultats du Modèle 2a : Concentration Chrome (9 classes)

Performance par algorithme		
Algorithme	Accuracy	F1
SVM (RBF)	90.60%	0.91
KNN	83.90%	0.84
Random Forest	83.67%	0.84
Gradient Boosting	83.29%	0.84

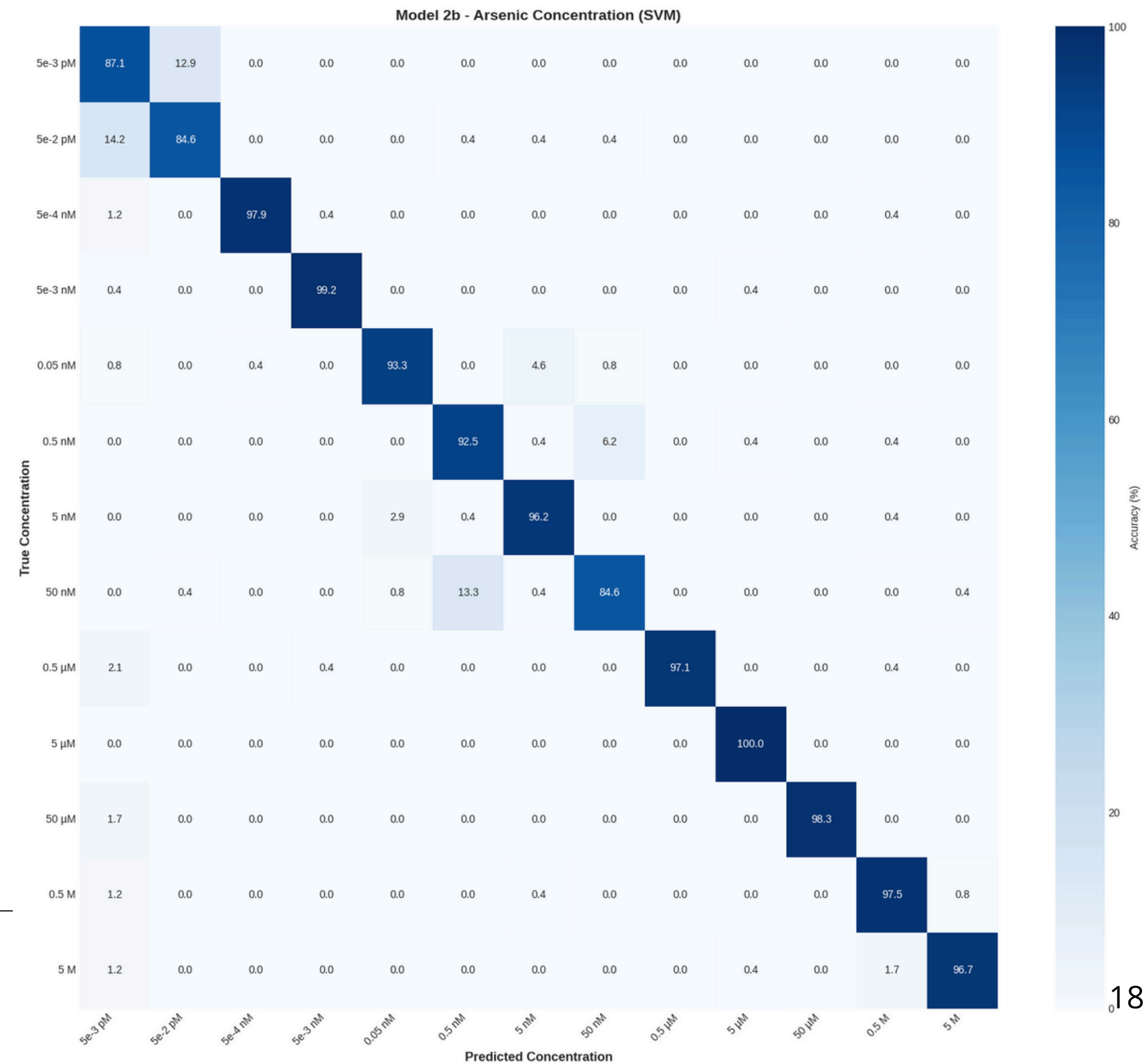
- Observations
- SVM nettement supérieur (+7%)
 - Confusions entre concentrations adjacentes
 - Extrêmes mieux classifiés



Résultats du Modèle 2b : Concentration Arsenic (13 classes)

Performance par algorithme		
Algorithme	Accuracy	F1
SVM (RBF)	94.23%	0.91
KNN	90.30%	0.90
Random Forest	90.22%	0.90
Gradient Boosting	89.96%	0.90

- Points positifs
- 🏆 Meilleure performance globale
 - ✓ 94% malgré 13 classes
 - ✓ Signatures As bien distinctes



Resume

Succès

- ✓ Tous les modèles $> 90\%$
- ✓ SVM optimal pour toutes les tâches
- ✓ Architecture hiérarchique validée

Détection contamination

90% de précision globale
pour identifier eau contaminée



Discussion et Perspectives

Analyse Critique : Points Forts et Limites

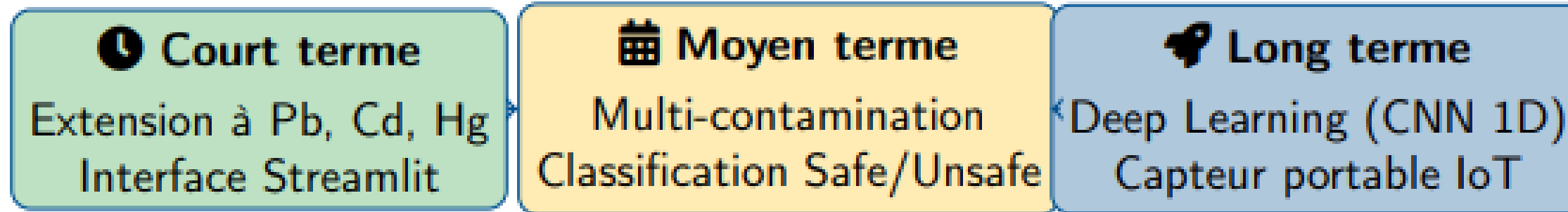
Points forts du système

- ✓ **Architecture modulaire** : chaque modèle optimisé pour sa tâche
- ✓ **Robustesse** : pipeline de prétraitement élimine le bruit
- ✓ **Interprétabilité** : décisions compréhensibles (vs. boîte noire)
- ✓ **Reproductibilité** : méthodologie documentée
- ✓ **Équilibrage SMOTE** : gestion efficace du déséquilibre

Limites identifiées

- ⚠ **Propagation d'erreurs** : erreur Modèle 1 → impacts Modèle 2
- ⚠ **Confusions adjacentes** : concentrations proches difficiles à distinguer
- ⚠ **Données laboratoire** : validation terrain non effectuée
- ⚠ **Mono-contamination** : pas de détection simultanée Cr+As

Perspectives d'Amélioration



🎯 **Objectif final** : Système de surveillance en temps réel pour l'agriculture





Conclusion

Conclusion

Contributions principales

- ① **Système IA de détection** combinant SERS et Machine Learning
- ② **Architecture hiérarchique** innovante à 3 modèles
- ③ **Pipeline complet** : prétraitement, SMOTE, classification
- ④ **Performances élevées** : 90-94% d'accuracy

Résultats clés

- ✓ Modèle 1 : **92%** (Type de métal)
- ✓ Modèle 2a : **91%** (Conc. Chrome)
- ✓ Modèle 2b : **94%** (Conc. Arsenic)

💡 Ce projet ouvre la voie à une surveillance intelligente de la qualité de l'eau

Merci pour votre attention !



Code disponible sur GitHub



arektoutmossab@gmail.com



QUESTIONS ?

Annexe : Bibliothèques Python

Catégorie	Bibliothèques
Manipulation données	NumPy, Pandas
Visualisation	Matplotlib, Seaborn
Prétraitement	SciPy
Machine Learning	Scikit-learn
Équilibrage	Imbalanced-learn (SMOTE)

Environnement

- Python 3.x
- Google Colab (GPU)
- Jupyter Notebook
- Visual Studio Code