### **Document de Synthèse : Recommandations et Architectures ML**

**Objectif :** Ce document résume les étapes de prétraitement clés issues de l'Analyse Exploratoire des Données (EDA) et propose plusieurs architectures de Machine Learning (combinants prétraitement et modèle) pour aborder le problème de classification des stations 5G (légitimes vs fausses).

**1. Synthèse des Recommandations de Prétraitement Issues de l'EDA**

L'analyse exploratoire a mis en lumière plusieurs caractéristiques importantes des données qui doivent guider le prétraitement :

* **Gestion du Déséquilibre des Classes :**
  + **Problème :** Forte sur-représentation de la classe 0 (~81%) par rapport aux classes 1 et 2 (~9.5% chacune).
  + **Solutions :**
    - Appliquer des techniques de **ré-échantillonnage** (ex: SMOTE pour oversampling, NearMiss pour undersampling) sur le jeu d'entraînement.
    - Utiliser la **pondération des classes** lors de l'entraînement du modèle.
    - Évaluer avec des **métriques adaptées** (F1-score pondéré/macro, Balanced Accuracy, AUC-PR, Matrice de Confusion).
* **Mise à l'Échelle des Features (Scaling) :**
  + **Problème :** Les features présentent des échelles et des variances différentes.
  + **Solution :** Appliquer une mise à l'échelle, par exemple avec StandardScaler, pour centrer et réduire les données. Entraîner le scaler *uniquement* sur les données d'entraînement.
* **Gestion de la Haute Dimensionalité :**
  + **Problème :** Nombre élevé de features (3456) par rapport aux échantillons (1491), risque de surajustement et "fléau de la dimension".
  + **Solutions :**
    - **Réduction de dimension :** Utiliser des techniques d'**extraction** (PCA avec suffisamment de composantes, Autoencodeurs) ou de **sélection** (basée sur variance, information mutuelle, importance via modèle).
    - **Régularisation :** Intégrer des techniques de régularisation fortes (L1, L2, Dropout) dans le modèle choisi.
* **Exploitation de la Structure Spatiale/Fréquentielle :**
  + **Observation :** Les données peuvent être vues comme une image 72x48, suggérant une structure locale potentiellement pertinente.
  + **Solution (si pertinent pour le modèle) :** **Remodeler (reshape)** les données d'entrée en format 2D (72x48) pour les modèles capables d'exploiter cette structure (ex: CNN).
* **Gestion des Valeurs Aberrantes (Outliers) :**
  + **Observation :** Présence potentielle de quelques valeurs extrêmes.
  + **Solution :** StandardScaler offre une certaine robustesse. Si nécessaire, RobustScaler ou des méthodes spécifiques de traitement des outliers peuvent être envisagées, mais ne sont peut-être pas indispensables en première approche.

**2. Architectures Possibles (Prétraitement + Modèle ML)**

Voici quelques pistes d'architectures combinant prétraitement et modèle, allant du plus classique au plus spécifique :

**Approche 1 : Classique avec Sélection/Réduction Forte**

* **Prétraitement :**
  1. Mise à l'échelle (StandardScaler).
  2. Gestion de l'imbalance (ex: SMOTE sur le train set *après* la séparation train/test).
  3. Réduction de dimension forte :
     + Option A : Sélection de features (ex: SelectKBest avec f\_classif ou mutual\_info\_classif, en choisissant un k pertinent via validation croisée).
     + Option B : PCA en conservant un nombre significatif de composantes (ex: expliquant 90-95% de la variance).
* **Modèle ML :**
  + Modèles d'ensemble robustes : **LightGBM**, **XGBoost**, **Random Forest**.
  + Ou **SVM** avec noyau non linéaire (ex: RBF).
* **Avantages :** Relativement rapide à entraîner, potentiellement plus interprétable (surtout avec sélection), moins de risque de surajustement grâce à la réduction forte.
* **Inconvénients :** Peut perdre de l'information utile lors de la réduction, pourrait ne pas capturer toutes les interactions complexes ou la structure 2D.

**Approche 2 : Réseaux de Neurones Denses (MLP)**

* **Prétraitement :**
  1. Mise à l'échelle (StandardScaler).
  2. Gestion de l'imbalance (ex: SMOTE ou pondération des classes dans la fonction de coût).
  3. (Optionnel) Réduction de dimension modérée (PCA ou Autoencodeur) si le MLP a du mal à converger ou surajuste trop.
* **Modèle ML :**
  + **Multi-Layer Perceptron (MLP)** : Plusieurs couches denses (Dense) avec activations non linéaires (ex: ReLU), **Batch Normalization** entre les couches, et **Dropout** pour la régularisation. La taille des couches et le taux de dropout sont à optimiser.
* **Avantages :** Capacité à modéliser des relations non linéaires complexes.
* **Inconvénients :** Moins interprétable ("boîte noire"), sensible aux hyperparamètres (architecture, optimiseur, taux d'apprentissage), ne tire pas parti de la structure 2D inhérente.

**Approche 3 : Réseaux de Neurones Convolutifs (CNN) - Exploitant la Structure 2D**

* **Prétraitement :**
  1. **Reshape** des données : Transformer chaque vecteur de 3456 features en une image 72x48 (avec 1 canal).
  2. Mise à l'échelle (StandardScaler appliqué *avant* le reshape ou normalisation type image [0,1] ou [-1,1] *après* le reshape).
  3. Gestion de l'imbalance (ex: SMOTE sur les données aplaties avant reshape, ou via un WeightedRandomSampler dans le DataLoader si utilisation de PyTorch/TensorFlow).
* **Modèle ML :**
  + **CNN 2D :** Séquence de :
    - Couches Conv2D (avec filtres de différentes tailles, ex: 3x3, 5x5).
    - Activation (ex: ReLU).
    - Batch Normalization.
    - MaxPooling2D.
    - Dropout.
  + Suivie d'une couche Flatten puis de couches Dense pour la classification finale (avec Dropout).
* **Avantages :** Conçu pour exploiter les structures locales/spatiales (ou fréquentielles ici), souvent très performant pour ce type de données structurées, la convolution agit comme une forme d'extraction de features pertinente.
* **Inconvénients :** Plus complexe à implémenter et optimiser, potentiellement plus gourmand en ressources (GPU recommandé), nécessite un réglage fin de l'architecture CNN.

**3. Considérations Générales pour Toutes les Approches**

* **Validation Croisée (Cross-Validation) :** Utilisez impérativement une validation croisée stratifiée (ex: StratifiedKFold) pour évaluer les performances de manière robuste et pour le réglage des hyperparamètres, étant donné la taille limitée du dataset et l'imbalance des classes.
* **Réglage des Hyperparamètres :** Explorez l'espace des hyperparamètres (ex: nombre de voisins pour SMOTE, k pour SelectKBest, nombre de composantes PCA, architecture du réseau, taux de dropout, paramètres de l'optimiseur, poids des classes) en utilisant des techniques comme GridSearchCV, RandomizedSearchCV ou l'optimisation Bayésienne, toujours en conjonction avec la validation croisée.
* **Pipeline :** Intégrez vos étapes de prétraitement et votre modèle dans un Pipeline (Scikit-learn) pour éviter les fuites de données (data leakage) lors de la validation croisée et simplifier le processus.

**Conclusion**

Il n'y a pas une unique "meilleure" architecture garantie. L'approche la plus prometteuse dépendra de l'expérimentation. L'approche **CNN (Approche 3)** semble théoriquement bien adaptée à la structure potentielle des données, mais les approches plus classiques **(Approche 1)** ou **MLP (Approche 2)** peuvent aussi donner d'excellents résultats avec un bon réglage et une gestion adéquate de la dimensionalité et de l'imbalance. Il est recommandé d'expérimenter au moins deux de ces approches et de comparer leurs performances via une validation croisée rigoureuse.