PROJET FIL ROUGE – VERSION 2

Système de décision de trading GBP/USD
(M1 → M15 → ML → RL → API → Docker)

Février 2026

Table des matières

1 Contexte général

Vous concevez un système de décision algorithmique sur la paire GBP/USD.
Fréquence brute : 1 minute
Fréquence décision : 15 minutes
À chaque décision :

— BUY

— SELL

— HOLD

Objectif : maximiser le profit cumulé sous contraintes réalistes :

— coûts de transaction

— drawdown limité

— robustesse inter-annuelle

— décisions mesurées

2 Données

2.1 Période disponible

— 2022

— 2023

— 2024

2.2 Split temporel obligatoire

Interdiction de split aléatoire.

— 2022 : Entraînement

— 2023 : Validation

— 2024 : Test final (jamais utilisé pour entraîner)

Walk-forward autorisé si documenté.

3 Structure imposée du projet

3.1 Phase 1 – Importation M1

— Fusion date + time → timestamp

— Vérification régularité 1 minute

— Tri chronologique

— Détection incohérences

3.2 Phase 2 – Agrégation M1 → M15

Aucune modélisation autorisée en M1.

Variable	Règle
open_15m	open 1ère minute
high_15m	max(high) sur 15 minutes
low_15m	min(low) sur 15 minutes
close_15m	close dernière minute

3.3 Phase 3 – Nettoyage M15

— Suppression bougies incomplètes
— Contrôle prix négatifs
— Détection gaps anormaux

4 Organisation du travail (mode sprint léger) et Git

4.1 Principe

Vous travaillez en mode **sprint léger** (sans Scrum formel) :
— vous **découpez** le projet en tâches claires,
— vous **répartissez** les tâches (1 ou 2 étudiants par groupe),
— vous **poussez** sur Git à chaque tâche terminée (pas de « gros push final »).

4.2 Règles Git obligatoires

— **Un dépôt Git par groupe**, avec historique lisible.
— **Une branche par tâche** (feature branch).
— **Chaque tâche doit apparaître sur Git** via commits réguliers.
— **Chaque étudiant doit pousser au moins une branche** (même en binôme).

4.3 Convention de nommage des branches

Objectif : que l'enseignant sache **qui** a poussé **quoi** et **pour quelle tâche**.

Format obligatoire :

`<prenomnom>__<Txx>__<mot-cle>`

où :
— `<prenomnom>` = identifiant court (ex : `aya`, `marc`, `ines`)
— `<Txx>` = numéro de tâche (ex : `T01`, `T06`)
— `<mot-cle>` = résumé court (ex : `m15_agg`, `features_pack`, `api_predict`)

Exemples :
— `aya__T01__import_m1`

— `ines__T02__m15_agg`

— `marc__T05__features_regime`

— `aya__T08__rl_env`

— `ines__T10__api_predict`

4.4 Convention de commits

Chaque commit doit décrire une action concrète.

Format recommandé :

`[Txx] verbe: description courte`

Exemples :

— `[T02] add: aggregation M1->M15`

— `[T05] fix: remove incomplete candles`

— `[T10] add: /predict endpoint with model_version`

4.5 Table des tâches (backlog minimal)

Chaque groupe doit remplir cette table **avant de coder** puis la mettre à jour.

ID	Tâche	Responsable	Branche Git
T01	Import M1 + contrôle régularité		
T02	Agrégation M1→M15		
T03	Nettoyage M15 + rapport qualité		
T04	Analyse exploratoire + ADF/ACF		
T05	Feature Pack V2 (court terme + régime)		
T06	Baseline règles + backtest simple		
T07	ML (split temporel + modèles + éval)		
T08	RL (env + reward + entraînement)		
T09	Évaluation robuste (benchmarks + 2024)		
T10	API (contrat + endpoints + chargement modèle)		
T11	Versioning modèle (v1/v2 + registry)		
T12	Docker + exécution reproductible		

5 Analyse exploratoire

Obligatoire :

— Distribution des rendements

— Volatilité dans le temps

— Analyse horaire
— Autocorrélation
— Test ADF

6 Feature Engineering – Version 2

Toutes les features sont calculées uniquement à partir du passé.

6.1 Bloc court terme

— return_1
— return_4
— ema_20
— ema_50
— ema_diff
— rsi_14
— rolling_std_20
— range_15m
— body
— upper_wick
— lower_wick

6.2 Bloc Contexte & Régime

Tendance long terme

— ema_200
— distance_to_ema200
— slope_ema50

Régime de volatilité

— atr_14
— rolling_std_100
— volatility_ratio

Force directionnelle

— adx_14
— macd
— macd_signal

7 Baseline obligatoire

Avant ML ou RL :

— Stratégie règles fixes
— Stratégie aléatoire
— Buy & Hold

8 Machine Learning

Objectif : prédire le mouvement de la prochaine bougie.

$$y = \begin{cases} 1 & \text{si } close_{t+1} > close_t \\ 0 & \text{sinon} \end{cases}$$

Exigences :

— Split temporel strict
— Modèle baseline
— Comparaison modèles
— Métriques statistiques et financières

9 Reinforcement Learning

9.1 Conception obligatoire sur papier

Avant codage :

1. Problème métier (objectif, contraintes, horizon)
2. Données (qualité, alignement, coûts)
3. State (features, normalisation, warm-up)
4. Action (discret ou allocation)
5. Reward (PnL ou PnL ajusté risque)
6. Environnement (simulateur, slippage, transaction cost)
7. Choix algorithme (justification obligatoire)

9.2 Paramètres clés

Paramètres de définition

state, action, reward, horizon, coûts

Paramètres d'entraînement

— γ
— learning rate
— exploration ϵ
— batch size
— epochs
— seed

Paramètres d'évaluation

— split temporel
— walk-forward
— Sharpe
— drawdown
— stress tests

10 Évaluation finale

Comparaison obligatoire :

— Random
— Règles
— ML
— RL

Métriques :

— Profit cumulé
— Maximum drawdown
— Sharpe simplifié
— Profit factor

Un modèle est valide uniquement s'il est robuste sur 2024.

11 Industrialisation

Architecture minimale :

```
project/
|
+-- data/
+-- features/
+-- models/
|   +-- v1/
|   +-- v2/
```

```
+-- training/
+-- evaluation/
+-- api/
+-- docker/
```

Règles :

— L'API expose uniquement le meilleur modèle.

— L'utilisateur ne peut pas relancer l'entraînement.

— Versioning modèle obligatoire.

— L'API charge automatiquement la version validée.

Message clé

Un modèle performant n'est pas celui qui gagne le plus sur 2022.
C'est celui qui :

— survit au changement de régime

— tient compte des coûts

— évite l'overfitting temporel

— est reproductible

— est industrialisable