HilmarCorp R&D Division

Note de développement R&D — Janvier 2025

Mémoire du marché : premières expérimentations séquentielles sur Bitcoin

Clovis Hilmarcher

Sommaire

Introduction

En janvier 2025, la HilmarCorp R&D Division a conduit une première série d'expérimentations séquentielles sur le Bitcoin, dans le cadre de ses travaux sur la mémoire de marché et les régimes de volatilité. L'objectif était de documenter, sur un actif unique mais liquide, l'existence éventuelle de dépendances temporelles dans l'enchaînement des états de volatilité, et d'évaluer dans quelle mesure ces régimes peuvent être modélisés de façon exploitable pour des briques futures de filtrage du risque et de risk overlay.

Ce travail s'inscrit dans la continuité de la littérature sur les propriétés "non i.i.d." des marchés financiers. Depuis les premiers résultats empiriques sur le volatility clustering (périodes de calme et d'agitation qui se succèdent par paquets) et les modèles de type GARCH, jusqu'aux modèles à changements de régime ou Hidden Markov Models appliqués aux actions, indices ou taux, de nombreux travaux ont montré que la volatilité suit des dynamiques de régimes plutôt qu'un bruit blanc homogène. La présente note reprend cette intuition dans un cadre volontairement restreint : un seul actif (BTC), à fréquence journalière, avec un label binaire de régime de volatilité et des architectures relativement simples, afin de mesurer de manière contrôlée ce que "voit" réellement un modèle séquentiel sur ce type de données.

Plus précisément, on cherche ici à répondre à une question opérationnelle : sur Bitcoin, entre 2017 et 2025, une architecture de type LSTM (Long Short-Term Memory) disposant d'un historique de 60 jours apporte-t-elle, pour la classification de régimes de volatilité future, un signal supplémentaire par rapport à des modèles instantanés sans mémoire (régression logistique, perceptron multicouche) appliqués au vecteur d'indicateurs à la date t ? Autrement dit, existe-t-il, à cette granularité, une mémoire de régime exploitable au-delà de la simple "photographie" des indicateurs techniques normalisés (rendements récents, volatilité réalisée, écarts aux moyennes, volume et participation) ?

Pour isoler cette contribution séquentielle, les modèles sont entraînés sur des données journalières de BTC couvrant la période 2017–2025, enrichies d'un ensemble cohérent d'indicateurs techniques et normalisées de manière strictement causale. Deux baselines non séquentielles : une régression logistique régularisée et un MLP peu profond fournissent un point de comparaison direct pour juger de l'apport (ou non) du LSTM en termes de classification de régimes, de qualité probabiliste (log-perte, AUC) et de diagnostics de risque (matrices de confusion, courbes ROC, précision–rappel).

Les sections suivantes présentent d'abord la construction du label de régime et des features (Section 2), puis le protocole expérimental et les métriques d'évaluation (Section 3). La Section 4 discute les résultats empiriques obtenus sur les baselines et le LSTM, en incluant diagnostics graphiques, études d'ablation et un exemple illustratif de filtrage de risque. Enfin, la Section 5 propose une lecture critique des résultats, discute la portée du cadre expérimental et esquisse des pistes de V2, avant de conclure sur le positionnement de ces travaux dans le programme de recherche de HilmarCorp.

I Données et préparation

1.1 Sources des données

L'expérimentation s'appuie sur des données de marché quotidiennes du Bitcoin, couvrant la période du 17 août 2017 au 10 novembre 2025. Cet horizon a été choisi de manière à inclure plusieurs cycles complets du marché crypto : bull, bear et phases latérales offrant ainsi un cadre idéal pour étudier la persistance ou la rupture des régimes comportementaux.

Les données ont été collectées directement via l'API officielle de Binance, principal marché spot du Bitcoin depuis 2017. Ce choix s'explique par la profondeur de son carnet d'ordres, la continuité historique de ses cotations et la granularité horaire des informations disponibles, qui en font une source de référence pour les travaux quantitatifs sur le BTC. L'utilisation de l'API, plutôt que de fichiers agrégés tiers, garantit une traçabilité intégrale du pipeline de collecte et la reproductibilité complète des résultats, conformément aux standards de recherche de HilmarCorp R&D Division.

Chaque observation journalière issue de l'API comprend :

- le prix d'ouverture (open),
- le prix le plus haut (high),
- le prix le plus bas (low),
- le prix de clôture (close),
- le volume total échangé sur la journée, exprimé en BTC,
- ainsi qu'un horodatage précis en fuseau UTC.

Les séries ont ensuite été réindexées sur un calendrier civil continu, assurant une progression temporelle uniforme et la préservation des séquences.
Aucune interpolation n'a été appliquée : les journées absentes ont été explicitement conservées comme manquantes afin de ne pas altérer la dépendance temporelle du signal. Les volumes extrêmes, souvent liés à des consolidations API ou des anomalies de carnet, ont été détectés par contrôle inter-quantile symétrique (±3σ) et ajustés sans lissage des prix.

L'ensemble des données brutes ainsi nettoyées : prix, volumes et métadonnées temporelles constitue la base empirique des expérimentations séquentielles menées dans cette étude. C'est à partir de cette fondation que seront ensuite dérivés les indicateurs techniques et les régimes de marché nécessaires à l'analyse de la mémoire temporelle grâce à un modèle séquentiel.

II Indicateurs dérivés (Feature Engineering)

2.1 contexte

Pour analyser la mémoire potentielle du marché et préparer l'entraînement d'un modèle séquentiel, il est nécessaire de disposer d'un ensemble d'indicateurs capables de capturer les dimensions essentielles du comportement du Bitcoin. L'objectif n'est pas d'optimiser un vecteur de features, mais de constituer un espace de variables cohérent, économiquement interprétable et suffisamment riche pour permettre au modèle de détecter d'éventuelles dépendances temporelles.

Dans cette étude, nous retenons un ensemble réduit mais représentatif d'indicateurs dérivés des séries OHLCV journalières. Ils couvrent quatre grandes dimensions du marché :

- Direction et Momentum : rendements logarithmiques multi-horizons, oscillateurs mesurant la persistance locale du mouvement ;
- Structure de tendance : moyennes mobiles relatives, indicateurs d'orientation (MACD) et de force directionnelle (ADX, DI^+/DI^-) ;
- Volatilité et compression : volatilité réalisée, largeur normalisée des bandes de Bollinger ;
- Participation : volumes relatifs, flux monétaires (MFI) et structure des chandeliers.

Ces indicateurs ne sont pas utilisés comme outils décisionnels, mais comme proxies standardisés d'états de marché, permettant de tester si un modèle séquentiel est capable d'en extraire une structure temporelle cohérente. L'enjeu de cette première expérimentation n'est pas la performance prédictive, mais la mise en évidence, ou non d'une forme de mémoire dans l'évolution des régimes de volatilité et de tendance.

En travaillant sur données journalières, l'ambition est de capturer des signaux persistants plutôt que des fluctuations micro-structurelles. Les features retenues sont construites de manière causale, normalisées et alignées temporellement, de manière à fournir un espace d'entrée propre au réseau LSTM et adapté à l'étude de la continuité et des transitions de régimes.

2.2 Indicateur technique

2.2.1 Direction

La dimension « Direction » regroupe les variables utilisées pour caractériser l'évolution immédiate du prix et la persistance locale du mouvement. Les rendements multi-horizons mesurent l'amplitude et la continuité du déplacement du prix sur différentes fenêtres temporelles, tandis que les oscillateurs de momentum décrivent l'intensité relative des gains et des pertes dans le court terme. Ces mesures ne visent pas à formuler une prévision directionnelle, mais à fournir une représentation normalisée des dynamiques élémentaires nécessaires à l'analyse séquentielle des régimes de marché.

A) Rendements multi-horizons (1, 5, 10, 30 jours)

Pour un prix de clôture P_t nous utilisons les rendements logarithmiques : additifs dans le temps et statistiquement plus stables sur longues fenêtres :

$$r^{(k)}t = \ln\left(\frac{P_t}{Pt-k}\right), \quad k \in \{1, 5, 10, 30\}.$$

En complément, nous suivons le positionnement relatif du prix sur sa moyenne mobile 30 jours :

$$\pi_t = \frac{P_t}{\mathrm{MA30}(P)t} - 1, \qquad \mathrm{MA30}(P)t = \frac{1}{30}\sum i = 0^{29} Pt - i.$$

Implémentation utilisée :

```
12  # --- Rendements & prix relatifs ---
13  df["ret_1d"] = np.log(df["close"]).diff()
14  df["ret_5d"] = df["close"].pct_change(5)
15  df["ret_10d"] = df["close"].pct_change(10)
16  df["ret_30d"] = df["close"].pct_change(30)
17
18  df["price_norm_30d"] = df["close"] / df["close"].rolling(30).mean() - 1
```

B) Momentum local (RSI, Stochastique, MACD)

Les indicateurs de Momentum quantifient la vitesse et l'essoufflement du mouvement dans le court terme. Ils permettent d'identifier les phases d'accélération, les retournements progressifs et les situations d'excès par rapport au comportement récent du prix. Dans une perspective séquentielle, ces oscillateurs fournissent des signaux normalisés sur la dominance acheteuse ou vendeuse instantanée et constituent un complément naturel aux rendements multi-horizons pour décrire la dynamique locale de marché.

B.1 RSI(14)

Le RSI mesure le ratio des gains/pertes lissés sur 14 jours (lissage de Wilder) ; borné entre 0 et 100, il sert de proxy normalisé du Momentum court terme : des excursions prolongées au-dessus (ou au-dessous) de 50 signalent la domination des gains (ou des pertes) et renseignent sur le régime : tendance ou retour à la moyenne.

$$\mathrm{RSI14}(t) = 100 - \frac{100}{1 + RS_t}, \qquad RS_t = \frac{\mathrm{EMA14}(\Delta^+ P)t}{\mathrm{EMA14}(\Delta^- P)t},$$

Où

$$\Delta^+ P = \max(\Delta P, 0), \ \Delta^- P = \max(-\Delta P, 0), \ \Delta P = P_t - Pt - 1.$$

Implémentation utilisée :

```
32    df["rsi_14"] = ta.momentum.RSIIndicator(df["close"], window=14).rsi()
```

B.2 Stochastique %K / %D (14, 3)

Le stochastique positionne la clôture dans le range récent ; borné entre 0 et 100, il sert de proxy de pression relative : des séjours prolongés en zone haute (ou basse) signalent une domination acheteuse (ou vendeuse) et renseignent sur tendance, cassure ou marché en range.

$$\%K_t = 100 \cdot \frac{C_t - L_t^{(14)}}{H_t^{(14)} - L_t^{(14)}}, \qquad \%D_t = \mathrm{SMA3}(\%K)t,$$

Avec

$$H^{(14)}t = \max i \in [0, 13]Ht - i, \ L^{(14)}t = \min i \in [0, 13]Lt - i.$$

Implémentation utilisée :

```
35    df["stoch_k"] = ta.momentum.StochasticOscillator(
36        high=df["high"], low=df["low"], close=df["close"], window=14).stoch()
37    df["stoch_d"] = ta.momentum.StochasticOscillator(
38        high=df["high"], low=df["low"], close=df["close"], window=14).stoch_signal()
```

B.3 MACD (12, 26, 9)

Le MACD compare deux moyennes mobiles exponentielles du prix : une rapide, très réactive aux variations récentes, et une lente, qui reflète la tendance de fond. Leur écart est

ensuite lissé par une ligne signal ; l'histogramme visualise ce différentiel : il s'étire quand le momentum accélère, se contracte ou bascule sous zéro quand le mouvement s'essouffle ou inverse.

$$\mathrm{MACD}t = \mathrm{EMA12}(P)t - \mathrm{EMA26}(P)_t, \quad \mathrm{Signal}t = \mathrm{EMA9}(\mathrm{MACD})_t, \quad \mathrm{Hist}_t = \mathrm{MACD}_t - \mathrm{Signal}_t.$$

Implémentation utilisée :

```
40    macd = ta.trend.MACD(df["close"])
41    df["macd"] = macd.macd()
42    df["macd_signal"] = macd.macd_signal()
43    df["macd_hist"] = macd.macd_diff()
```

2.2.2 Structure de tendance

La structure de tendance regroupe les indicateurs visant à mesurer l'orientation générale du marché et la force relative des mouvements. Contrairement aux oscillateurs de momentum, centrés sur les fluctuations locales, ces mesures s'intéressent à la cohérence d'un mouvement prolongé et à son degré d'organisation. Elles permettent de distinguer les phases de tendance établie, les périodes de consolidation et les situations de marché en range, éléments essentiels dans un cadre d'analyse séquentielle.

A) Moyennes mobiles (écarts normalisés 10, 20, 50, 200 jours)

Les moyennes mobiles servent ici à mesurer le niveau de portage du prix par rapport à sa tendance de fond. Plutôt que d'utiliser les niveaux bruts, nous suivons l'écart relatif du prix de clôture P_t à sa moyenne mobile simple sur n jours, ce qui fournit une mesure plus stationnaire et moins redondante avec le prix.

Pour une moyenne mobile simple sur *n* jours :

$$\mathrm{MA}n(P)t = \frac{1}{n}\sum i = 0^{n-1} Pt - i, \qquad n \in \{10, 20, 50, 200\},$$

l'écart normalisé est défini par :

$$\delta^{(n)}t = \frac{P_t}{\mathrm{MA}n(P)_t} - 1.$$

Un $\delta_t^{(n)}$ durablement positif traduit un portage haussier (prix au-dessus de sa moyenne), tandis qu'un $\delta_t^{(n)}$ négatif et persistant indique un biais baissier.

Implémentation utilisée :

```
55    # --- Moyennes mobiles (ratios) ---
56    df["ma_10"] = df["close"] / df["close"].rolling(10).mean() - 1
57    df["ma_20"] = df["close"] / df["close"].rolling(20).mean() - 1
58    df["ma_50"] = df["close"] / df["close"].rolling(50).mean() - 1
59    df["ma_200"] = df["close"] / df["close"].rolling(200).mean() - 1
```

B) CCI(20)

Le Commodity Channel Index mesure l'écart normalisé du prix typique à sa moyenne récente. Il sert ici à détecter les phases de sur- / sous-extension autour d'un régime de tendance : des valeurs élevées (ou très basses) et persistantes signalent un mouvement organisé, tandis que les retours rapides vers la zone neutre traduisent plutôt un rééquilibrage.

On définit d'abord le prix typique :

$$TP_t = \frac{H_t + L_t + C_t}{3},$$

Puis la moyenne mobile simple et la déviation moyenne sur 20 jours :

$$\mathrm{MA20}(TP)t = \frac{1}{20}\sum i = 0^{19} TPt - i, \qquad MD_{20}(TP)t = \frac{1}{20}\sum i = 0^{19} \ |TP_{t-i} - \mathrm{MA}_{20}(TP)_t \ |.$$

Le CCI(20) est alors donné par :

$$\mathrm{CCI20}(t) = \frac{TP_t - \mathrm{MA20}(TP)t}{0,015 \times MD20(TP)_t}.$$

Des niveaux extrêmes et durables de CCI_{20} indiquent une extension marquée du prix par rapport à son équilibre local, utile pour qualifier l'état du trend (mature, en extension, en normalisation).

Implémentation utilisée :

```python
    df["cci_20"] = ta.trend.CCIIndicator(
        high=df["high"],
        low=df["low"],
        close=df["close"],
        window=20
    ).cci()
```

C) ADX(14), DI⁺, DI⁻

L' Average Directional Index (ADX) mesure la force d'une tendance, indépendamment de son sens, tandis que DI^+ et DI^- en décrivent l'orientation. Ensemble, ils permettent de distinguer les phases de trend organisé des périodes de marché en range/chop, point crucial pour un modèle séquentiel.

On part des mouvements directionnels positifs et négatifs :

$$+DM_t = \max(H_t - H_{t-1}, 0), \qquad -DM_t = \max(L_{t-1} - L_t, 0),$$

et de la True Range :

$$TR_t = \max\{H_t - L_t,\ |H_t - C_{t-1}|,\ |L_t - C_{t-1}|\}.$$

Après lissage de type Wilder sur 14 jours, on définit :

$$DI_t^+ = 100 \times \frac{\mathrm{EMA}(+DM)_t}{\mathrm{EMA}(TR)_t}, \qquad DI_t^- = 100 \times \frac{\mathrm{EMA}(-DM)_t}{\mathrm{EMA}(TR)_t},$$

$$DX_t = 100 \times \frac{|DI_t^+ - DI_t^-|}{DI_t^+ + DI_t^-}, \qquad ADX_t = \mathrm{EMA}(DX)_t.$$

- Un ADX élevé indique un mouvement structuré (haussier si $DI^+ > DI^-$, baissier sinon).
- Un ADX faible signale un marché peu directionnel, où la volatilité est moins "portée" par un trend.

Dans notre cadre, ces variables servent à distinguer les segments où la mémoire séquentielle porte un trend clair de ceux où le marché oscille autour d'un équilibre local.

Implémentation utilisée :

```
adx14 = ta.trend.ADXIndicator(
    high=df["high"],
    low=df["low"],
    close=df["close"],
    window=14
)
df["adx_14"]  = adx14.adx()
df["adx_pos"] = adx14.adx_pos()  # DI+
df["adx_neg"] = adx14.adx_neg()  # DI-
```

2.2.3 Volatilité et compression

La volatilité et la compression de prix décrivent l'amplitude des mouvements et la façon dont l'incertitude se concentre ou se détend dans le temps. Là où la structure de tendance s'intéresse à la direction du mouvement, ces indicateurs capturent l'intensité du risque porté par chaque régime. Ils permettent d'identifier les phases de volatilité élevée et persistante, les épisodes de contraction ("volatility squeeze") et les transitions entre régimes calmes et stressés, éléments clés pour un modèle séquentiel.

A) Volatilité réalisée (7, 30 jours)

La volatilité réalisée mesure la dispersion observée des rendements sur une fenêtre glissante. Elle fournit une estimation non paramétrique du risque effectif porté par le marché, en agrégeant les chocs de prix récents.

À partir des rendements logarithmiques journaliers

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right),$$

La volatilité réalisée sur *k* jours est définie par :

$$\sigma_{k,t} = \sqrt{\frac{1}{k}\sum_{i=0}^{k-1}(r_{t-i} - \bar{r}_{k,t})^2}, \qquad k \in \{7, 30\},$$

Où $\bar{r}_{k,t}$ désigne la moyenne des rendements sur la même fenêtre.

Dans cette étude, $\sigma_{7,t}$ capture la volatilité courte, sensible aux chocs récents, tandis que $\sigma_{30,t}$ décrit le niveau de risque de fond du régime courant.

Implémentation utilisée :

```
63    # --- Volatilité ---
64    df["vol_7d"] = df["ret_1d"].rolling(7).std()
65    df["vol_30d"] = df["ret_1d"].rolling(30).std()
```

B) Bandes de Bollinger : largeur normalisée

Les bandes de Bollinger quantifient la dispersion du prix autour de sa moyenne mobile, en combinant information de tendance et de volatilité. Dans notre cas, nous retenons uniquement la largeur relative des bandes, utilisée comme proxy de compression / expansion du marché.

Pour une moyenne mobile $\mathrm{MA}_{20}(P)t$ et un écart-type $\sigma 20, t$ sur 20 jours, les bandes classiques s'écrivent :

$$\mathrm{BB}^{\mathrm{haut}}t = \mathrm{MA20}(P)t + 2\,\sigma 20, t, \qquad \mathrm{BB}^{\mathrm{bas}}t = \mathrm{MA20}(P)t - 2\,\sigma 20, t.$$

Nous suivons la largeur normalisée :

$$w_t = \frac{\mathrm{BB}_t^{\mathrm{haut}} - \mathrm{BB}^{\mathrm{bas}}t}{\mathrm{MA20}(P)_t},$$

qui mesure l'ouverture relative du "canal" de prix.

Un w_t faible indique une phase de compression (range étroit, volatilité contenue), souvent associée à des régimes calmes ou à des phases de pré-rupture. À l'inverse, un w_t durablement élevé traduit une expansion de volatilité, typique des régimes directionnels intenses ou des épisodes de stress.

Implémentation utilisée :

```
70    bb = BollingerBands(close=df["close"], window=20, window_dev=2)
71    df["boll_width"] = bb.bollinger_wband()
```

2.2.4 Participation

La participation de marché regroupe les indicateurs liés aux flux échangés et à la micro-structure des chandeliers. Là où la direction et la tendance décrivent le mouvement du prix, ces mesures renseignent sur l'intensité des échanges et la façon dont le prix se forme à l'intérieur de chaque séance. Elles sont particulièrement utiles pour distinguer les phases de marché "portées" par un flux significatif de celles où les variations de prix se produisent sur des volumes faibles ou déséquilibrés.

A) Volume relatif 20 jours

Le volume relatif à 20 jours mesure l'intensité des échanges du jour par rapport à un niveau de référence récent. Il sert ici de proxy simple de "sur-activité" ou de "désert de liquidité", deux configurations souvent associées à des transitions de régime ou à des phases de capitulation/exubérance.

On définit d'abord la moyenne mobile simple des volumes sur 20 jours :

$$\overline{V}_{20}(t) = \frac{1}{20}\sum_{i=0}^{19} V_{t-i},$$

puis le volume relatif :

$$\text{vol\_rel\_20d}(t) = \frac{V_t}{\overline{V}_{20}(t)}.$$

Un $\text{vol\_rel\_20d} \gg 1$ signale une séance anormalement active, tandis qu'une valeur durablement inférieure à 1 traduit un environnement de liquidité réduite. Ce type d'information permet d'identifier les phases de marché marquées par un afflux ou un retrait de participation, souvent associées aux changements d'équilibre entre acheteurs et vendeurs.

Implémentation utilisée :

```
87    df["vol_rel_20d"] = df["volume"] / df["volume"].rolling(20).mean()
```

B) Money Flow Index (MFI 14)

Le Money Flow Index combine prix et volume pour estimer la pression d'achat ou de vente sur une fenêtre donnée. Contrairement à un volume brut, il pondère les échanges par

la direction des mouvements de prix, ce qui en fait un proxy de “flux directionnel” plutôt que de simple activité.

Le MFI sur 14 jours est calculé à partir du prix typique TP_t (moyenne de H_t, L_t et C_t) des volumes et des flux monétaires positifs/négatifs. Il est borné entre 0 et 100 :

- des valeurs élevées et persistantes indiquent une pression acheteuse dominante ;
- des valeurs faibles signalent une pression vendeuse prolongée.

Le MFI_{14} renseigne ainsi sur l’intensité directionnelle sous-jacente aux mouvements observés : il met en évidence les phases où les flux monétaires soutiennent une tendance, ainsi que celles où une divergence prix-volume apparaît souvent précurseur d’un affaiblissement ou d’une inversion du mouvement.

Implémentation utilisée :

```
90    df["mfi_14"] = ta.volume.MFIIndicator(
91        high=df["high"],
92        low=df["low"],
93        close=df["close"],
94        volume=df["volume"],
95        window=14
96    ).money_flow_index()
```

C) Ratios de chandeliers (corps / ombres)

Les ratios de chandeliers décrivent la façon dont le prix a évolué à l’intérieur de la séance : part du mouvement réalisée dans le corps (open → close) versus les extrêmes atteints (high/low). Ils fournissent une information fine sur la structure intrajournalière, sans recourir à une granularité infra-day.

On définit :

- le corps relatif :

$$\mathrm{candle\_body}_t = \frac{C_t - O_t}{H_t - L_t + \varepsilon},$$

- l'ombre supérieure relative :

$$\text{upper\_shadow}_t = \frac{H_t - \max(C_t, O_t)}{H_t - L_t + \varepsilon},$$

- l'ombre inférieure relative :

$$\text{lower\_shadow}_t = \frac{\min(C_t, O_t) - L_t}{H_t - L_t + \varepsilon},$$

Avec ε un terme de stabilisation numérique.

Un corps large associé à des ombres réduites traduit un mouvement directionnel affirmé au cours de la séance. À l'inverse, des ombres longues et un corps restreint reflètent davantage le rejet des extrêmes, l'hésitation ou des épisodes de micro-volatilité. Ces variables sont utiles pour distinguer les journées de conviction nette des configurations plus neutres ou techniques.

Implémentation utilisée :

```
100   df["candle_body"] = (df["close"] - df["open"]) / (df["high"] - df["low"] + 1e-8)
101   df["upper_shadow"] = (df["high"] - np.maximum(df["close"], df["open"])) / (df["high"] - df["low"] + 1e-8)
102   df["lower_shadow"] = (np.minimum(df["close"], df["open"]) - df["low"]) / (df["high"] - df["low"] + 1e-8)
```

III Méthodologie expérimentale

3.1 Construction du label de régime

A) Contexte

Les marchés financiers et plus particulièrement le Bitcoin ne se comportent jamais comme un processus homogène dans le temps. Ils évoluent par régimes successifs, chacun caractérisé par un niveau distinct d'incertitude, de dispersion des prix et de pression directionnelle.

Dans la littérature quantitative, cette organisation temporelle se traduit par un phénomène robuste : la volatilité ne se distribue pas uniformément mais se regroupe en blocs cohérents, souvent appelés volatility clusters.

Certaines phases se déroulent sous un régime d'équilibre local, où les variations quotidiennes demeurent contenues autour de leur norme récente ; d'autres basculent dans un régime d'expansion du risque, marqué par une amplification soudaine et persistante de la dispersion des rendements.

Ces transitions ne sont ni aléatoires, ni purement erratiques : elles portent une structure temporelle que les modèles séquentiels (LSTM) sont précisément conçus pour capturer.

L'objectif du label est donc de formaliser cette segmentation naturelle du marché en construisant une mesure opérationnelle capable d'indiquer si, à une date t, le système se situe dans un environnement de risque "normalisé" ou dans une phase où le niveau d'incertitude anticipé excède significativement sa référence historique.

Ce choix est motivé par une considération opérationnelle : il formalise la distinction entre un environnement de risque "normalisé" et des phases où la volatilité anticipée dépasse significativement son niveau de référence. Un marché "calme" n'est pas celui où il ne se passe rien, mais celui où les fluctuations demeurent proportionnées à leur histoire récente.

À l'inverse, un marché "tendu" est celui où les chocs futurs attendus dépassent la capacité du marché à absorber ces variations, signalant potentiellement l'arrivée d'un changement d'équilibre, d'un stress latent, ou d'une transition de régime.

B) Définition opérationnelle du label

La définition du label exploite exclusivement des objets déjà établis dans la section 2 : rendements logarithmiques et volatilité réalisée. L'idée centrale est d'évaluer si la volatilité réalisée dans les jours à venir dépasse significativement la volatilité de fond estimée sur une fenêtre longue.

La construction se déroule en trois temps :

1. Mesure de la volatilité future, obtenue en calculant la dispersion réalisée des rendements dans l'horizon $[t+1, t+FWD\_HORIZON]$. . Cette mesure représente la quantité d'incertitude que le marché s'apprête à délivrer immédiatement après t.

2. Mesure de la volatilité de référence, calculée sur une fenêtre plus longue (30 jours), qui reflète le niveau de risque "normalisé" du régime courant.

3. Comparaison des deux niveaux via un ratio :

$$R_t = \frac{\text{vol}_{\text{future}}}{\text{vol}_{\text{reference}}}$$

Lorsque ce ratio excède 1.20, le marché bascule dans un régime considéré comme significativement plus volatil que sa norme récente. Ce seuil de +20 % est cohérent avec les discontinuités empiriques observées sur BTC : la majorité des épisodes de stress se manifeste par une rupture nette de cette magnitude. Le label reflète ainsi une structure de volatilité observée dans les données, plutôt qu'une segmentation arbitraire imposée a priori.

Le label obtenu est volontairement binaire : non pas pour simplifier, mais pour stabiliser l'analyse séquentielle et éviter les ambiguïtés propres aux partitions multi-classes. Il constitue un objet propre, robuste, interprétable, et surtout parfaitement adapté à l'étude de la mémoire temporelle.

Implémentation utilisée

```python
def build_regime_labels(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    df = df.sort_values("timestamp").reset_index(drop=True)

    # Rendements log (déjà utilisés dans les features)
    df["ret_1d"] = np.log(df["close"]).diff()

    # Volatilité future sur 3 jours
    fwd_vol = (
        df["ret_1d"]
        .shift(-1)
        .rolling(FWD_HORIZON)
        .std()
    )
    df["fwd_vol_3d"] = fwd_vol.shift(-(FWD_HORIZON - 1))

    # Volatilité de référence (30 jours)
    df["ref_vol_30d"] = df["ret_1d"].rolling(REF_WINDOW).std()

    # Ratio futur / normal
    df["vol_ratio"] = df["fwd_vol_3d"] / df["ref_vol_30d"]

    # Nettoyage
    df = df.replace([np.inf, -np.inf], np.nan)
    df = df.dropna(subset=["fwd_vol_3d", "ref_vol_30d", "vol_ratio"]).reset_index(drop=True)

    # Label final
    df["regime"] = (df["vol_ratio"] > VOL_RATIO_THRESHOLD).astype(int)

    return df
```

3.2 Normalisation des features

Les indicateurs décrits précédemment présentent des échelles et des distributions hétérogènes : rendements logarithmiques potentiellement non bornés, mesures de volatilité,

écarts normalisés aux moyennes mobiles, oscillateurs bornés entre 0 et 100, proxies de flux et de participation. Avant l'entraînement des modèles (régressions de base, MLP, LSTM), il est nécessaire de ramener ces variables dans un espace numérique cohérent, afin d'éviter qu'une famille de features ne domine artificiellement les autres du seul fait de son échelle.Dans cette première expérimentation, nous retenons une approche simple et standard : chaque indicateur est centré et réduit individuellement sur l'échantillon d'étude.

Pour une variable x_t^j (colonne *j* du jeu de données), on calcule :

- la moyenne empirique μ^j sur l'horizon considéré ;
- l'écart-type empirique σ^j.

Puis on construit la version normalisée :

$$\tilde{x}_t^j = \frac{x_t^j - \mu^j}{\sigma^j}.$$

Cette transformation est appliquée colonne par colonne à l'ensemble des features utilisées dans les modèles, sans toucher à la variable cible (le régime) ni aux horodatages. Elle a trois effets principaux :

- aligner les ordres de grandeur des différentes familles d'indicateurs (direction, tendance, volatilité, participation) ;
- stabiliser la dynamique des gradients lors de l'optimisation des modèles neuronaux ;
- limiter le poids arbitraire des variables naturellement plus volatiles (rendements, volatilité réalisée, proxies de volume).

Dans le cadre de ce travail exploratoire, la normalisation est réalisée en deux temps : les moyennes et écarts-types sont estimés une fois pour toutes sur le seul bloc d'entraînement, puis appliqués telles quelles aux données de validation. Les modèles de base (régression logistique, MLP) et le modèle séquentiel utilisent ainsi exactement le même vecteur de features normalisées, ce qui permet de comparer de manière homogène leurs performances tout en respectant strictement la causalité temporelle.

Implémentation utilisée :

```python
from sklearn.preprocessing import StandardScaler

feature_cols = [
    "ret_1d", "ret_5d", "ret_10d", "ret_30d",
    "price_norm_30d",
    "rsi_14", "stoch_k", "stoch_d",
    "macd", "macd_signal", "macd_hist",
    "cci_20", "adx_14", "adx_pos", "adx_neg",
    "vol_7d", "vol_30d", "boll_width",
    "ma_10", "ma_20", "ma_50", "ma_200",
    "vol_rel_20d", "mfi_14",
    "candle_body", "upper_shadow", "lower_shadow"
]

scaler = StandardScaler()
df[feature_cols] = scaler.fit_transform(df[feature_cols])
```

3.3 Mise en forme séquentielle des données (fenêtrage LSTM)

A) Contexte

Les modèles séquentiels et en particulier les architectures de type LSTM (Long Short-Term Memory) ne consomment pas des observations indépendantes, mais des *segments temporels* structurés.
L'objectif n'est plus d'expliquer un label à partir d'un vecteur instantané, mais d'identifier la cohérence interne d'une trajectoire, ses ruptures, et les motifs réguliers qui précèdent un changement de régime.

Pour cela, les données normalisées issues des sections précédentes doivent être transformées en un tenseur tridimensionnel :

$$\mathcal{X} \in \mathbb{R}^{N \times T \times F}$$

où :

- **N** : nombre total de séquences exploitables,
- **T** : longueur de la fenêtre temporelle (lookback),
- **F** : nombre total de features normalisées.

Cette opération appelée *sliding window transformation* fait passer la table chronologique bidimensionnelle classique :

à un objet structuré contenant, pour chaque date t, l'historique complet nécessaire au LSTM :

$$\mathbf{x}_t^{(seq)} = \begin{bmatrix} x_{t-T+1}^1 & x_{t-T+1}^2 & \cdots & x_{t-T+1}^F \\ x_{t-T+2}^1 & x_{t-T+2}^2 & \cdots & x_{t-T+2}^F \\ \vdots & \vdots & & \vdots \\ x_t^1 & x_t^2 & \cdots & x_t^F \end{bmatrix}$$

Ce format respecte strictement la causalité : aucune information future n'est injectée dans la séquence d'entrée.

B) Justification du choix du lookback

Le choix de la fenêtre temporelle Test un paramètre crucial : une fenêtre trop courte manque de contexte, une fenêtre trop longue dilue l'information pertinente et augmente le bruit. Ici, l'objectif n'est pas de prédire un prix mais d'analyser la *mémoire des régimes de volatilité*.

La littérature empirique sur BTC montre que :

- les cycles de compression → explosion de volatilité s'étalent généralement sur 15 à 40 jours**,**
- les transitions de tendance structurelle se forment sur 1 à 3 mois,
- les clusters de volatilité persistante ont une demi-vie comprise entre 8 et 20 jours.

Un horizon de T = 60 jours s'imposent donc comme un compromis robuste :

- suffisamment long pour capter les débuts et fins de clusters,
- suffisamment court pour éviter la sur-dispersion et le "vanishing information",
- cohérent avec les pratiques des desks volatility / derivatives en quant research.

Ce choix améliore aussi la stabilité numérique du LSTM en lui offrant des séquences riches mais non dégénérées.

C) Construction formelle des séquences

Soit une série chronologique normalisée :

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T, ..., \mathbf{x}_n\} \quad \text{o} \quad \mathbf{x}_t \in \mathbb{R}^F.$$

Pour chaque date $t \geq T$, on définit une séquence :

$$\mathbf{S}_t = (\mathbf{x}_{t-T+1}, ..., \mathbf{x}_t) \in \mathbb{R}^{T \times F}.$$

Le label associé est simplement :

$$y_t = \text{regime}(t)$$

avec garantie que :

$$\mathbf{S}_t \quad \text{ne contient aucune information postrieure} \quad t.$$

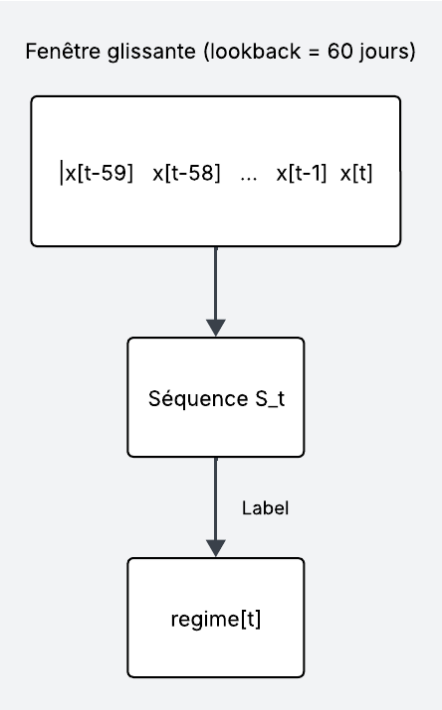
D) Précautions causales et alignement

Afin d'éviter tout leakage temporel :

- Le label à prédire est positionné sur la fin de la fenêtre, jamais à l'intérieur.
- Aucun indicateur technique n'utilise de données futures, tous sont calculés en backward-looking.
- La normalisation est effectuée avant le fenêtrage, mais exclusivement sur la zone d'entraînement.
- Le split train/val/test est strictement chronologique, garantissant la validité empirique du protocole.
- Les NaN initiaux liés aux rolling windows sont supprimés avant construction des séquences.

Ces contraintes rapprochent l'expérience des pratiques industrielles, où la causalité opérationnelle est impérative (risk, execution, derivatives modelling).

E) Schéma explicatif



Implémentation utilisée :

```python
class RegimeDataset(Dataset):
    def __init__(self, df: pd.DataFrame, feature_cols, seq_len: int):
        self.df = df
        self.feature_cols = feature_cols
        self.seq_len = seq_len

    def __len__(self):
        return len(self.df) - self.seq_len

    def __getitem__(self, idx):
        seq = self.df.loc[idx:idx + self.seq_len - 1, self.feature_cols].values
        y = int(self.df.loc[idx + self.seq_len, "regime"])
        return (
            torch.tensor(seq, dtype=torch.float32),
            torch.tensor(y, dtype=torch.long),
        )
```

3.4 Modèles étudiés

A) Contexte

Sur la base du jeu de données construit dans les sections précédentes (rendements journaliers, indicateurs dérivés normalisés, label binaire de régime fondé sur la volatilité réalisée), trois familles de modèles de classification sont considérées, toutes entraînées sous le même protocole temporel. L'objectif n'est pas d'optimiser une performance prédictive absolue, mais de tester empiriquement l'existence d'une mémoire exploitable dans l'enchaînement des régimes de marché.

Deux grandes classes de modèles sont mises en regard.

La première regroupe des modèles instantanés sans mémoire explicite : une régression logistique et un perceptron multicouche (MLP). Pour ces modèles, chaque observation est décrite par un vecteur de variables explicatives normalisées $x_t \in \mathbb{R}^F$ (rendements, volatilité réalisée, écarts aux moyennes mobiles, indicateurs de participation, etc.) ; l'ordre dans lequel ces configurations de marché se succèdent n'est jamais pris en compte. La seconde classe repose sur un modèle séquentiel à mémoire explicite de type LSTM, qui reçoit en entrée, pour chaque date t, une trajectoire normalisée de 60 jours $(x_{t-T+1}, \ldots, x_t)$ et peut, en principe, exploiter la structure temporelle des transitions de régime.

Dans ce cadre, la régression logistique joue le rôle de référence linéaire pleinement interprétable : elle teste l'existence d'une séparation affine entre régimes dans l'espace des indicateurs. Le MLP étend ce dispositif en autorisant des combinaisons non linéaires des mêmes variables, tout en restant strictement amnésique : il ne voit jamais l'historique, seulement l'instantané du vecteur x_t . Le LSTM, enfin, occupe un rôle différent : il ne se limite pas à raffiner une frontière de décision dans $\mathbb{R}^F$, mais cherche à quantifier l'apport spécifique de la dynamique séquentielle en modélisant directement l'évolution des trajectoires sur 60 jours.

La comparaison entre ces modèles est conduite à protocole égal : mêmes variables normalisées, même label de régime, même découpage chronologique train/validation, même horizon d'étude. Toute différence de performance ne peut donc provenir que de la nature de l'information effectivement consommée – instantanée pour les baselines, séquentielle pour le LSTM. Les sous-sections suivantes détaillent la formulation et le rôle méthodologique des baselines non séquentielles, qui serviront de point d'ancrage pour l'interprétation des résultats du LSTM.

B) Régression logistique (baseline linéaire)

La régression logistique constitue le point de départ le plus simple pour la classification des régimes. Le modèle ne cherche pas à exploiter l'historique des trajectoires, mais uniquement la configuration du vecteur x_t à la date *t*. Chaque journée est ainsi résumée par un vecteur normalisé $y_t \in \{0,1\}$ indiquant si le marché se trouve, immédiatement après *t*, dans un régime de volatilité future "normalisée". $(y_t = 0)$ ou "tendue" $(y_t = 1)$ au sens de la section 3.1.

La régression logistique associe à chaque observation x_t une probabilité conditionnelle d'être en régime tendu de la forme :

$$\mathbb{P}(y_t = 1 \mid x_t) = \sigma(w^\top x_t + b) = \frac{1}{1 + \exp\left(-(w^\top x_t + b)\right)},$$

Où $w \in \mathbb{R}^F$ est le vecteur de coefficients et $b \in \mathbb{R}$ un biais scalaire. La frontière de décision correspond au demi-espace défini par $w^\top x_t + b = 0$ dans l'espace des indicateurs : un changement de signe de cette quantité se traduit par un basculement de probabilité de part et d'autre du seuil ½

Les paramètres (w, b) sont estimés par maximum de vraisemblance régularisé. En notant $\hat{p}_t = \mathbb{P}(y_t = 1 \mid x_t)$ la probabilité prédite et $\mathcal{D}_{\text{train}}$ l'ensemble des indices d'entraînement, la fonction de coût considérée est :

$$\mathcal{L}(w, b) = -\sum_{t \in \mathcal{D}_{\text{train}}} [y_t \log \hat{p}_t + (1 - y_t) \log(1 - \hat{p}_t)] + \lambda \|w\|_2^2,$$

Avec $\lambda > 0$ un paramètre de régularisation L2. Le terme quadratique sur w limite la sensibilité du modèle à des co-mouvements spécifiques de certaines variables – notamment

celles liées à la direction et à la volatilité, naturellement plus volatiles – et impose une structure plus parcimonieuse aux coefficients.

Dans ce dispositif, la régression logistique joue deux rôles distincts. D'une part, elle offre une référence interprétable : chaque composante de *w* peut être lue comme un poids directionnel porté par une variable donnée (rendement court terme, volatilité réalisée, volume relatif, etc.), ce qui permet d'identifier les indicateurs qui contribuent le plus, de manière linéaire, à la séparation entre régimes. D'autre part, elle fournit une borne inférieure "sans mémoire" sur ce qu'il est possible d'expliquer à partir du seul instantané des indicateurs, sans aucune information sur l'ordre dans lequel les états de marché se succèdent.

Les performances de cette baseline linéaire sur l'échantillon de validation servent de point de comparaison direct pour évaluer l'apport des modèles non linéaires (MLP) et, surtout, du modèle séquentiel LSTM : toute amélioration significative devra être interprétée à la lumière de l'information supplémentaire exploitée (non-linéarité instantanée ou dynamique temporelle).

Implémentation utilisée :

En pratique, la régression logistique est implémentée via la classe LogisticRegression de la bibliothèque *scikit-learn*, avec pénalisation L2 par défaut et optimisation par maximum de vraisemblance régularisé. Les variables explicatives sont préalablement centrées-réduites, et le découpage train/validation est strictement chronologique, identique à celui utilisé pour les autres modèles. Le code correspondant est le suivant :

```
107  def run_baselines(df, feature_cols, train_ratio=0.8):
108      X = df[feature_cols].values
109      y = df["regime"].values
110      split_idx = int(train_ratio * len(df))
111      X_tr, X_val = X[:split_idx], X[split_idx:]
112      y_tr, y_val = y[:split_idx], y[split_idx:]
113
114      results = {}
115
116      logreg = LogisticRegression(max_iter=200)
117      logreg.fit(X_tr, y_tr)
118      results["logistic"] = logreg.score(X_val, y_val)
119      print(f"[BASELINE] logistic val_acc={results['logistic']:.4f}")
120
```

C) Perceptron multicouche (MLP)

Le perceptron multicouche (MLP) prolonge la régression logistique en autorisant des transformations non linéaires du vecteur de variables explicatives. Là où le modèle linéaire ne peut séparer les régimes que par un hyperplan dans l'espace des indicateurs, le MLP introduit des couches cachées apprenant des combinaisons non linéaires des mêmes

signaux. La structure temporelle reste cependant inchangée : comme pour la régression logistique, chaque observation est traitée indépendamment, sur la base de x_t uniquement.

On considère toujours pour chaque date *t* un vecteur normalisé $x_t \in \mathbb{R}^F$ et un label binaire $y_t \in \{0,1\}$. Dans cette étude, on retient une architecture simple à deux couches cachées entièrement connectées. En notation compacte, le réseau réalise la transformation suivante :

$$h_t^{(1)} = \phi\,(W^{(1)} x_t + b^{(1)}),$$

$$h_t^{(2)} = \phi\,(W^{(2)} h_t^{(1)} + b^{(2)}),$$

$$\hat{p}_t = \sigma\,(w^{(3)\top} h_t^{(2)} + b^{(3)}),$$

où : $W^{(1)} \in \mathbb{R}^{64\times F}$ et $W^{(2)} \in \mathbb{R}^{32\times 64}$ sont les matrices de poids des couches cachées, $w^{(3)} \in \mathbb{R}^{32}$ le vecteur de poids de la couche de sortie, $b^{(1)}, b^{(2)}, b^{(3)}$ les biais correspondants, $\phi(\cdot)$ la sigmoïde logistique qui renvoie une probabilité $\hat{p}_t \in (0,1)$ d'être en régime tendu.

Les paramètres du réseau

$$\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, w^{(3)}, b^{(3)}\}$$

sont estimés en minimisant une log-perte binaire régularisée sur l'échantillon d'entraînement :

$$\mathcal{L}(\theta) = -\sum_{t\in\mathcal{D}_{\text{train}}} [\,y_t \log \hat{p}_t + (1-y_t)\log(1-\hat{p}_t)\,] + \lambda \sum_k \|W^{(k)}\|_F^2,$$

où le second terme contrôle la norme des poids des couches cachées (régularisation L2) afin de limiter le sur-apprentissage sur des configurations rares de variables. L'optimisation est réalisée par descente de gradient stochastique (type Adam dans l'implémentation scikit-learn), avec itérations successives jusqu'à convergence numérique ou jusqu'à un nombre maximal d'epochs. L'entraînement s'effectue sur les mêmes variables normalisées et avec le même découpage chronologique train/validation que pour la régression logistique, de façon à garantir une comparaison cohérente.

Dans ce cadre, le MLP occupe trois fonctions méthodologiques complémentaires. Il fournit d'abord une baseline non linéaire sans mémoire : en le comparant à la régression logistique, on mesure ce que l'on gagne simplement en passant d'un classificateur linéaire à un modèle capable de capturer des interactions complexes entre indicateurs – par exemple

une configuration de type “volatilité réalisée élevée + volume relatif anormalement fort + MACD en extension” – tout en restant aveugle à la structure séquentielle. Il sert ensuite de test de richesse du vecteur de variables : si le MLP n’améliore pas significativement la classification par rapport à la régression logistique, cela suggère que l’essentiel de l’information pertinente est capté de manière quasi linéaire, et que le bénéfice potentiel d’un modèle séquentiel résidera principalement dans la mémoire temporelle plutôt que dans la non-linéarité statique. Enfin, le MLP constitue un point de comparaison direct pour le LSTM : les deux modèles partagent exactement le même espace de variables normalisées, mais seul le LSTM observe l’historique complet $(x_{t-T+1}, \ldots, x_t)$. L’écart de performance entre les deux renseigne donc sur la contribution propre de la dynamique séquentielle par rapport à une simple projection non linéaire de l’instantané.

Implémentation utilisée :

Le MLP est implémenté via la classe MLPClassifier de *scikit-learn*, avec deux couches cachées de tailles 64 et 32 neurones, fonction d’activation ReLU, pénalisation L2 par défaut et un maximum de 500 itérations d’optimisation. Les mêmes matrices $X_{\text{train}}, X_{\text{val}}$ et vecteurs de labels $y_{\text{train}}, y_{\text{val}}$ que pour la régression logistique sont utilisés. Le code est le suivant :

```
123     mlp = MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=500, random_state=42)
124     mlp.fit(X_tr, y_tr)
125     results["mlp"] = mlp.score(X_val, y_val)
126     print(f"[BASELINE] mlp       val_acc={results['mlp']:.4f}")
```

D) Réseau séquentiel à mémoire explicite (LSTM)

L’approche séquentielle introduit une différence conceptuelle majeure par rapport aux modèles précédents. Ni la régression logistique ni le MLP n’ont accès à l’ordre temporel dans lequel les configurations de marché se succèdent : chaque état x_t est traité isolément, comme si l’enchaînement des régimes n’avait aucune structure exploitable. Or l’organisation empirique de la volatilité (clusters persistants, phases de compression puis d’expansion, transitions progressives entre épisodes de calme et de stress) suggère qu’une forme de dépendance temporelle peut être porteuse d’information. $(x_{t-T+1}, \ldots, x_t)$. Cette mémoire n’est pas une simple moyenne mobile : elle résulte d’un mécanisme récursif qui pondère dynamiquement ce qu’il convient de retenir, d’oublier ou de mettre à jour.

Le LSTM (Long Short-Term Memory) introduit précisément une mémoire interne capable d’intégrer, pour chaque observation, non seulement l’état courant x_t , mais aussi la trajectoire historique $(x_{t-T+1}, \ldots, x_t)$. Cette mémoire n’est pas une simple moyenne mobile : elle résulte d’un mécanisme récursif qui pondère dynamiquement ce qu’il convient de retenir, d’oublier ou de mettre à jour.

Formellement, à partir d'une séquence normalisée $(x_{t-T+1}, \ldots, x_t)$, le LSTM maintient un état caché h_k et un état mémoire c_k mis à jour à chaque pas $k \in \{t-T+1, \ldots, t\}$ via :

$$
\begin{aligned}
f_k &= \sigma(W_f x_k + U_f h_{k-1} + b_f) && \text{(porte d'oubli)} \\
i_k &= \sigma(W_i x_k + U_i h_{k-1} + b_i) && \text{(porte d'injection)} \\
\tilde{c}_k &= \tanh(W_c x_k + U_c h_{k-1} + b_c) && \text{(candidate mémoire)} \\
c_k &= f_k \odot c_{k-1} + i_k \odot \tilde{c}_k && \text{(mise à jour du réservoir)} \\
o_k &= \sigma(W_o x_k + U_o h_{k-1} + b_o) && \text{(porte de sortie)} \\
h_k &= o_k \odot \tanh(c_k) && \text{(projection séquentielle)}
\end{aligned}
$$

où $\sigma(\cdot)$ est une sigmoïde logistique, $\odot$ le produit élément-par-élément, et les matrice $(W_f, U_f, \ldots)$ les paramètres appris lors de l'entraînement. Le réseau prétend alors à une probabilité d'appartenance au régime tendu via :

$$\hat{p}_t = \sigma\left(w^\top h_t + b\right)$$

Les paramètres $\theta = \{W_\bullet, U_\bullet, b_\bullet, w, b\}$ sont estimés par minimisation d'une log-perte binaire régularisée, comme pour le MLP, mais appliquée sur des séquences :

$$\mathcal{L}(\theta) = -\sum_{t \in \mathcal{D}_{train}} \left[y_t \log \hat{p}_t + (1 - y_t) \log(1 - \hat{p}_t) \right] + \lambda \|\theta\|_2^2$$

L'entraînement s'effectue sans aucun chevauchement entre validation et entraînement, et avec une normalisation strictement établie sur l'horizon d'apprentissage, afin d'éviter tout leakage temporel lié au calcul des statistiques de normalisation. La comparaison avec la régression logistique et le MLP est réalisée à protocole égal : mêmes features, même label, même split chronologique, même critère de validation. La seule source de différence réside donc dans l'accès éventuel au passé $(x_{t-T+1}, \ldots, x_t)$.

Cette distinction structurelle place le LSTM dans un rôle méthodologique précis : il ne cherche pas à obtenir la meilleure métrique de classification possible, mais à quantifier l'existence d'une dépendance exploitable dans la dynamique de transition entre régimes. Si les performances du LSTM dépassent significativement celles du MLP, cela indique que l'information pertinente n'est pas contenue uniquement dans la configuration instantanée des indicateurs, mais également dans la trajectoire qui les relie, confirmant empiriquement la présence d'une mémoire séquentielle dans l'organisation des régimes de volatilité.

Implémentation utilisée :

Dans ces premières expériences, le modèle séquentiel est implémenté en PyTorch via un module RegimeLSTM. La partie récurrente est constituée d'une couche LSTM à 128 unités cachées (hidden_size = 128, num_layers = 1, batch_first = True), alimentée par des séquences de longueur 60 construites comme en 3.3.

La sortie de la couche LSTM (état caché à la dernière date de la fenêtre) est ensuite projetée par un petit réseau entièrement connecté de type

$$> h_t \in \mathbb{R}^{128} \xrightarrow{W_1, b_1} \mathbb{R}^{64} \xrightarrow{\text{ReLU}} \mathbb{R}^{64} \xrightarrow{W_2, b_2} \mathbb{R}^2, >$$

dont les deux composantes correspondent aux logits des classes “régime normalisé” (0) et “régime tendu” (1).

L’optimisation est réalisée avec l’algorithme Adam (taux d’apprentissage fixé à 10^{-3}, sur des mini-batches de 32 séquences, en minimisant une entropie croisée binaire standard (fonction CrossEntropyLoss de PyTorch), sans pondération explicite des classes. Le run de référence présenté en section 4 repose sur 40 époques d’apprentissage successives, avec suivi systématique de la log-perte et de l’accuracy sur l’échantillon de validation à chaque epoch.

```
44  class RegimeLSTM(nn.Module):
45      def __init__(self, input_dim, hidden_dim=128):
46          super().__init__()
47          self.lstm = nn.LSTM(
48              input_size=input_dim,
49              hidden_size=hidden_dim,
50              num_layers=1,
51              batch_first=True,
52          )
53          self.fc = nn.Sequential(
54              nn.Linear(hidden_dim, 64),
55              nn.ReLU(),
56              nn.Linear(64, 2),
57          )
58
59      def forward(self, x):
60          out, (h_n, _) = self.lstm(x)
61          last_h = h_n[-1]
62          return self.fc(last_h)
```

3.5 Protocole d’entraînement et d’évaluation

L’ensemble des modèles est entraîné sous un protocole commun, de manière à ce que les différences de performance reflètent uniquement la nature de l’information utilisée (instantanée vs séquentielle), et non des choix de calibration hétérogènes. Le protocole

contrôle successivement : le découpage temporel, la gestion du déséquilibre de classes, la procédure d'optimisation et les métriques d'évaluation.

A) Découpage temporel et respect de la causalité

Le jeu de données séquentiel construit en 3.3 est scindé en deux blocs chronologiques contigus :

- Entraînement : premier bloc couvrant environ 80 % des observations, du 17 août 2017 jusqu'à une date de coupure t_{split} ;
- Validation : bloc résiduel couvrant les 20 % de dates restants, de $t_{\text{split}} + 1$ jusqu'au 10 novembre 2025.

Une séquence $(x_{t-T+1}, \ldots, x_t)$ appartient à un unique bloc, déterminé par sa dernière date t. Les statistiques de normalisation sont recalculées exclusivement sur le bloc d'entraînement, puis appliquées telles quelles au bloc de validation. Aucun recouvrement n'est donc possible : ni au niveau des labels, ni au niveau des features.

Dans ces premières expériences, aucun bloc de test distinct n'est encore utilisé : toute l'analyse hors-échantillon repose sur le segment de validation, afin de concentrer les observations disponibles sur l'estimation des régimes et la comparaison des architectures.

B) Gestion du déséquilibre de classes

Le label binaire défini en 3.1 est par construction asymétrique : sur l'horizon étudié, les régimes de volatilité élevée (classe 1) représentent une fraction minoritaire des observations (de l'ordre d'un tiers), la classe "calme" (0) restant dominante.

Dans ces premières expériences, on ne modifie pas la fonction de perte pour refléter cet équilibre : la régression logistique, le MLP et le LSTM sont tous entraînés en minimisant une log-perte binaire standard (entropie croisée) sur le bloc d'apprentissage, sans pondération explicite des classes.

Le déséquilibre est pris en compte au niveau de l'évaluation, via des métriques robustes au déséquilibre via des métriques robustes au déséquilibre, en particulier la balanced accuracy, le rappel et le F1-score de la classe tendue, ainsi que l'inspection détaillée des matrices de confusion. L'objectif n'est donc pas de "compenser" le déséquilibre dans la loss, mais de le rendre explicite dans la lecture des performances.

C) Procédure d'optimisation et critères d'arrêt

Pour chaque modèle, l'entraînement est conduit en minimisant la log-perte pondérée sur le bloc d'apprentissage, avec sélection des itérations par surveillance de la performance sur la validation :

- la régression logistique et le MLP sont optimisés via les solveurs standard de scikit-learn (maximisation de vraisemblance régularisée pour la régression, descente de gradient stochastique type Adam pour le MLP), avec un nombre maximal d'itérations fixé à 500 ;

- le LSTM est entraîné sur des mini-batches de séquences (taille de batch 32), en minimisant la même log-perte binaire standard, avec l'optimiseur Adam (taux d'apprentissage 10^{-3}). Dans le run de référence, le modèle est entraîné pendant 40 époques successives ; la log-perte et l'accuracy sont monitorées à chaque epoch sur l'échantillon de validation, comme illustré par les courbes de la section 4.2. Aucun mécanisme d'early stopping n'est activé à ce stade : l'objectif est de documenter pleinement la dynamique d'overfitting du modèle séquentiel. Dans tous les cas, aucune information du bloc de test n'est utilisée à ce stade : le test reste strictement réservé à l'évaluation finale, une fois tous les hyperparamètres fixés.

D) Métriques d'évaluation

L'objectif des modèles n'est pas seulement de maximiser une précision globale, mais de mesurer finement la capacité à détecter les régimes de volatilité élevée, qui constituent la classe rare mais économiquement la plus critique. L'évaluation est donc conduite sur l'échantillon de validation à partir des probabilités prédites $\hat{p}_t = \mathbb{P}(y_t = 1 \mid \cdot)$ et des labels observés $y_t \in \{0,1\}$.

Pour un seuil de décision $\tau \in (0,1)$, on définit la prédiction binaire

$$\hat{y}_t = \mathbb{1}\{\hat{p}_t \geq \tau\}.$$

On retient $\tau = 0,5$ comme seuil neutre, afin de ne favoriser a priori ni la classe normale ni la classe tendue.

Sur l'ensemble des dates de validation $\mathcal{D}_{\text{val}}$, on construit la matrice de confusion :

- *TP* (true positives) : nombre de dates avec $y_t = 1$ et $\hat{y}_t = 1$;
- *FP* (false positives) : $y_t = 0$ et $\hat{y}_t = 1$;
- *TN (true negatives)* : $y_t = 0$ et $\hat{y}_t = 0$;
- *FN* (false negatives) : $y_t = 1$ et $\hat{y}_t = 0$.

À partir de ces quantités, plusieurs métriques complémentaires sont considérées :

Précision globale (accuracy)

$$\mathrm{Acc} = \frac{TP + TN}{TP + TN + FP + FN},$$

qui mesure la part d'observations correctement classées. Cette métrique est informative sur la performance moyenne, mais peut être trompeuse en présence de déséquilibre de classes.

Sensibilité au régime tendu (rappel de la classe 1)

$$\mathrm{Recall}_{(1)} = \frac{TP}{TP + FN},$$

qui quantifie la proportion de régimes de forte volatilité effectivement détectés. Économiquement, il s'agit de la capacité du modèle à signaler les épisodes de risque élevé.

Spécificité du régime normal (rappel de la classe 0)

$$\mathrm{Recall}_{(0)} = \frac{TN}{TN + FP},$$

qui mesure la capacité à ne pas déclencher de signal de stress lorsque le marché reste dans un régime de volatilité normalisée.

Balanced accuracy / macro-rappel

Pour neutraliser l'effet du déséquilibre, on suit la moyenne des rappels par classe :

$$\mathrm{BalAcc} = \frac{1}{2}\left(\mathrm{Recall}_{(0)} + \mathrm{Recall}_{(1)}\right).$$

Cette quantité donne à chaque régime le même poids, indépendamment de sa fréquence d'apparition.

F1-score sur le régime tendu

Le F1 met l'accent sur la détection correcte de la classe 1 en agrégeant précision et rappel :

$$\mathrm{Precision}_{(1)} = \frac{TP}{TP + FP}, \qquad F1_{(1)} = 2 \cdot \frac{\mathrm{Precision}_{(1)} \cdot \mathrm{Recall}_{(1)}}{\mathrm{Precision}_{(1)} + \mathrm{Recall}_{(1)}}.$$

Un $F1_{(1)}$ élevé indique que le modèle identifie les régimes tendus sans générer trop de faux signaux.

Qualité probabiliste : log-perte et AUC

Indépendamment du seuil τ, la qualité des probabilités $\hat{p}_t$ est évaluée via :

la log-perte binaire (déjà utilisée comme fonction d'entraînement) sur la validation,

$$\mathrm{LogLoss} = -\frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{t \in \mathcal{D}_{\mathrm{val}}} [y_t \log \hat{p}_t + (1 - y_t) \log(1 - \hat{p}_t)\,],$$

qui pénalise fortement les prédictions très confiantes mais erronées ;

L'aire sous la courbe ROC (AUC), obtenue en faisant varier τ et en traçant le couple (taux de faux positifs, taux de vrais positifs). Un AUC proche de 0,5 correspond à un tirage aléatoire, tandis qu'une valeur nettement supérieure traduit une bonne capacité de ranking entre régimes calmes et régimes tendus.

En complément, la courbe précision–rappel pour la classe tendue (régime 1), construite en faisant varier le seuil et en reportant la précision en fonction du rappel. Dans un contexte de déséquilibre de classes, ce diagnostic met directement en regard la proportion d'épisodes de forte volatilité effectivement détectés et la quantité de faux signaux générés.

En pratique, les résultats sont reportés pour un seuil fixe $\tau = 0,5$, ainsi que sous forme de courbes ROC et de tableaux de classification. L'analyse comparative entre les trois modèles (régression logistique, MLP, LSTM) se concentre principalement sur :

- le balanced accuracy et le *F1* pour juger la détection des régimes de risque élevé ;

- la log-perte et l'AUC, pour apprécier la qualité probabiliste des signaux.

Ces métriques fournissent un cadre homogène pour comparer les baselines instantanées et le modèle séquentiel, et pour isoler l'apport propre de la mémoire temporelle dans la classification des régimes de volatilité.

Enfin, au-delà de ces indicateurs scalaires, l'évaluation s'appuie sur une couche de diagnostics graphiques et d'expériences dérivées construits à partir des sorties quotidiennes du pipeline Python (timestamp, prix de clôture, label observé, prédiction discrète, probabilité $\hat{p}_t$). La section 4.4 exploite cette table de diagnostic pour représenter le prix du BTC coloré par régime observé/prédit, la trajectoire de $\hat{p}_t$ en regard du prix et du seuil 0,5 , ainsi que les corrélations de Pearson entre $\hat{p}_t$ et un sous-ensemble de variables explicatives clefs ; ces figures précisent visuellement où le modèle bascule en régime tendu et quels drivers dominent ce basculement. La section 4.5 s'appuie sur les mêmes métriques

(balanced accuracy, F1, log-perte, AUC) pour comparer des ablations ciblées du LSTM (suppression explicite des features de volatilité, réduction de la longueur de séquence) et quantifier la sensibilité du signal séquentiel à ces briques d'information. Enfin, la section 4.6 projette le signal $\hat{p}_t$ dans un cadre d'allocation simple : une stratégie jouet qui neutralise l'exposition lorsque $\hat{p}_t$ dépasse un seuil donné. Son equity et ses indicateurs de risque (rendement cumulé, volatilité annualisée, drawdown maximal, ratio de Sharpe) sont calculés de manière mécanique à partir des rendements quotidiens du BTC pondérés par l'exposition, puis comparés à un buy-and-hold passif ; l'exercice reste illustratif mais montre comment un filtre de régime probabiliste peut être intégré, sans hypothèses supplémentaires, dans des modèles de contrôle du risque plus complets.

IV Résultats empiriques

4.1 Performances globales des modèles

Les performances des trois familles de modèles sur l'échantillon de validation sont résumées dans le Tableau 1. Les métriques sont celles définies en section 3.5 (accuracy, balanced accuracy, rappel de la classe tendue, F1-score de la classe 1, log-perte et AUC), calculées avec un seuil de décision fixé à 0,5 sur la probabilité prédite. Les résultats correspondent au run de référence avec une fenêtre séquentielle de 60 jours et 40 époques d'apprentissage pour le LSTM.

Tableau 1 – Performances des modèles sur l'échantillon de validation

(seq_len = 60, epochs = 40)

Modèle	Accuracy	Balanced accuracy	Recall (classe 1)	F1 (classe 1)	Log-perte	AUC
Logistique	0,719	0,668	0,514	0,549	0,551	0,750
MLP	0,577	0,587	0,616	0,492	1,860	0,628
LSTM (60j)	0,535	0,526	0,500	0,416	3,378	0,532

Sur cet échantillon, la régression logistique s'impose comme une baseline solide. Avec une accuracy d'environ 0,72, une balanced accuracy proche de 0,67 et une AUC autour de 0,75, un classificateur strictement linéaire, appliqué au seul vecteur instantané d'indicateurs normalisés, parvient déjà à séparer de manière non triviale les régimes de volatilité "normalisée" et "tendue". Cela confirme qu'une part significative du signal de régime est effectivement encodée dans la configuration cross-sectionnelle des features (rendements récents, volatilité réalisée, écarts aux moyennes mobiles, participation), sans mobiliser la structure séquentielle.

Le perceptron multicouche (MLP) ne transforme pas cette base en gain massif de performance. Malgré sa capacité à modéliser des interactions non linéaires entre indicateurs, il n'améliore pas substantiellement l'AUC par rapport à la régression logistique et se dégrade nettement en log-perte. Ce profil est typique d'un modèle un peu plus expressif entraîné sur un volume de données limité : il s'adapte davantage à certaines configurations rares, mais cette flexibilité supplémentaire ne se traduit pas par une généralisation plus robuste. Dans ce cadre, le MLP joue surtout son rôle de test de richesse du vecteur de variables : le fait qu'il ne surpasse pas nettement la baseline linéaire suggère que, à fréquence journalière, l'essentiel de l'information exploitable sur le régime reste capturable par une combinaison quasi linéaire des indicateurs.

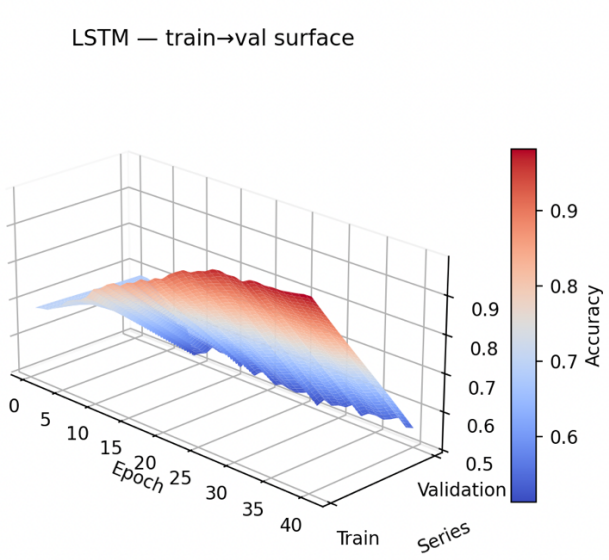
Le LSTM, enfin, introduit une mémoire explicite sur des séquences de 60 jours. Sur l'échantillon d'entraînement, le réseau atteint très rapidement des accuracies proches de 1, ce qui indique qu'il est parfaitement capable de reconstituer les labels à partir des trajectoires observées. En validation, en revanche, l'accuracy, la balanced accuracy et l'AUC retombent à des niveaux voisins de 0,53–0,54, tandis que la log-perte reste élevée. Ce décalage marqué entre apprentissage et validation est caractéristique d'un sur-apprentissage séquentiel : la dynamique que le LSTM a apprise sur l'historique d'entraînement ne se prolonge pas de manière stable hors-échantillon. Dans le protocole retenu (données journalières, label binaire de volatilité future, architecture LSTM simple), l'accès à l'historique complet $(x_{t-59}, \ldots, x_t)$ ne se traduit donc pas par un gain systématique par rapport aux baselines sans mémoire.

Pris ensemble, ces résultats ne remettent pas en cause l'hypothèse d'une mémoire de marché, mais en bornent la portée dans ce cadre précis. Ils indiquent que, sur Bitcoin et à cette granularité, une part non négligeable de l'information de régime est déjà contenue dans la photographie instantanée des indicateurs, et que la composante séquentielle exploitable par un LSTM vanilla reste, au mieux, de faible amplitude. Cette observation motive la suite du programme de recherche : affiner la définition du label, explorer des architectures séquentielles plus contraintes et tester des fréquences plus fines, afin de distinguer ce qui relève d'une véritable mémoire de régime de ce qui n'est qu'un artefact de construction de features.

4.2 Courbes d'apprentissage et stabilité de l'optimisation

Pour interpréter les résultats du Tableau 1, il est instructif d'examiner la dynamique d'apprentissage du modèle séquentiel. La Figure 4.2a représente, sous forme de surface, l'évolution de l'accuracy du LSTM sur l'échantillon d'entraînement et sur la validation au fil des époques.

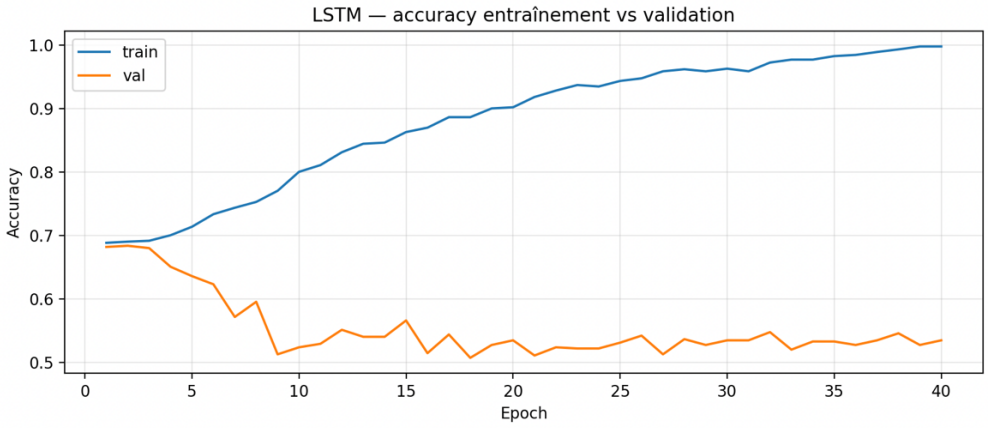
Figure 4.2a – Accuracy entraînement / validation du LSTM en fonction des epoch .



Cette visualisation met en évidence un écart croissant entre les deux séries : la "crête" rouge correspondant à l'accuracy d'entraînement progresse rapidement au-delà de 0,9, tandis que la partie "validation" de la surface reste nettement plus basse et tend vers un plateau autour de 0,52–0,55. Autrement dit, le réseau apprend à mémoriser très efficacement les séquences vues pendant l'entraînement, mais cette compétence ne se transpose pas de manière symétrique hors-échantillon.

La Figure 4.2b détaille ce constat en traçant séparément les courbes d'accuracy entraînement et validation. La courbe "train" croît quasi monotoniquement de ~0,69 à des valeurs proches de 1 sur 40 époques, confirmant la capacité du LSTM à reconstituer presque parfaitement les labels sur l'historique d'apprentissage. À l'inverse, la courbe "val" décroît progressivement depuis ~0,68 vers ~0,53, avant de se stabiliser dans une bande étroite autour de ce niveau.

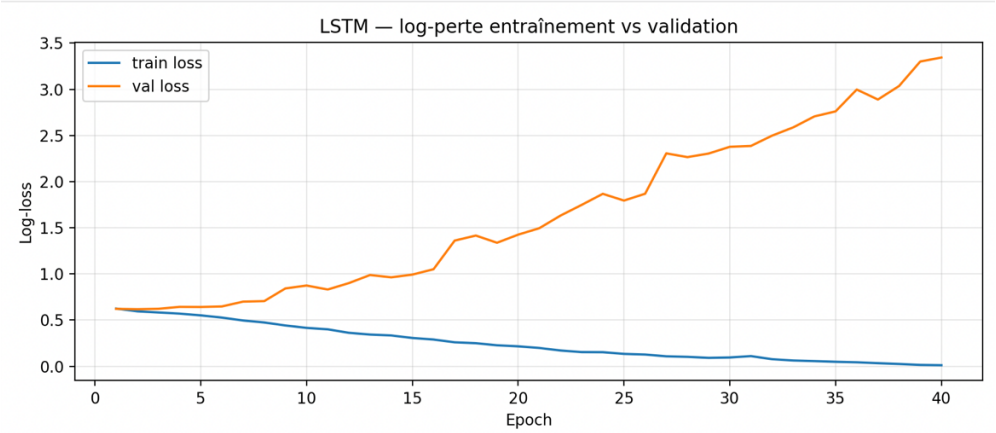
Figure 4.2b – Accuracy entraînement vs validation du LSTM (seq_len = 60, 40 époques)



Cette divergence entre accuracy entraînement et validation est typique d'un sur-apprentissage séquentiel : le modèle exploite sa forte capacité pour ajuster finement les trajectoires du jeu d'entraînement, mais la structure temporelle qu'il en déduit n'est pas suffisamment stable pour généraliser.

Les courbes de log-perte (Figure 4.2c) racontent la même histoire sous un angle probabiliste. La log-perte d'entraînement décroît régulièrement d'environ 0,6 jusqu'à des valeurs quasi nulles, signe que le réseau produit des probabilités très confiantes et globalement correctes sur les séquences vues. La log-perte de validation suit au contraire une trajectoire croissante, passant de ~0,6 à plus de 3,3 au fur et à mesure des époques : le LSTM devient de plus en plus sûr de prédictions hors-échantillon qui sont fréquemment erronées.

Figure 4.2c – Log-perte entraînement vs validation du LSTM (seq_len = 60, 40 epoch)



On observe ainsi un comportement numériquement sain (pas d'explosion de gradients, pas de divergence de la loss), mais mal régularisé : la complexité du LSTM est clairement excessive par rapport à la quantité d'information réellement exploitable à cette granularité journalière et pour ce label de volatilité future. En pratique, un schéma d'early stopping couperait l'apprentissage dès les premières époques, avant que la log-perte de validation n'ait commencé à se dégrader ; même dans cette fenêtre courte, cependant, le LSTM ne surpasse pas de façon robuste la régression logistique.

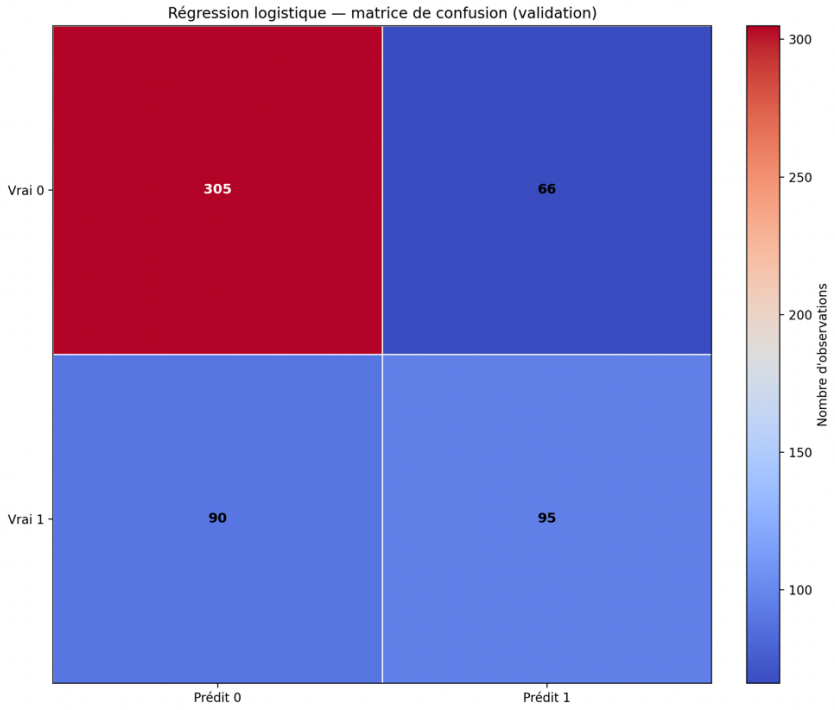
Pour la régression logistique et le MLP, l'optimisation scikit-learn (non reproduite ici pour ne pas alourdir la présentation) converge en quelques dizaines, respectivement quelques centaines d'itérations, avec une log-perte validation monotone décroissante et sans oscillations marquées. Les difficultés observées sur le modèle séquentiel ne proviennent donc pas d'un problème d'optimisation, mais d'un mismatch structurel entre la richesse du LSTM et le signal effectivement présent dans les données à ce niveau de résolution.

4.3 Diagnostics de classification par régime

Au-delà des scores agrégés du Tableau 1, l'enjeu est de comprendre comment les modèles se trompent : quelles dates de stress sont effectivement captées, lesquelles sont manquées, et à quel prix en faux signaux. Cette sous-section explore donc la structure fine de la classification par régime, en s'appuyant sur les matrices de confusion et sur le comportement des scores de probabilité lorsqu'on fait varier le seuil de décision.

A) Régression logistique

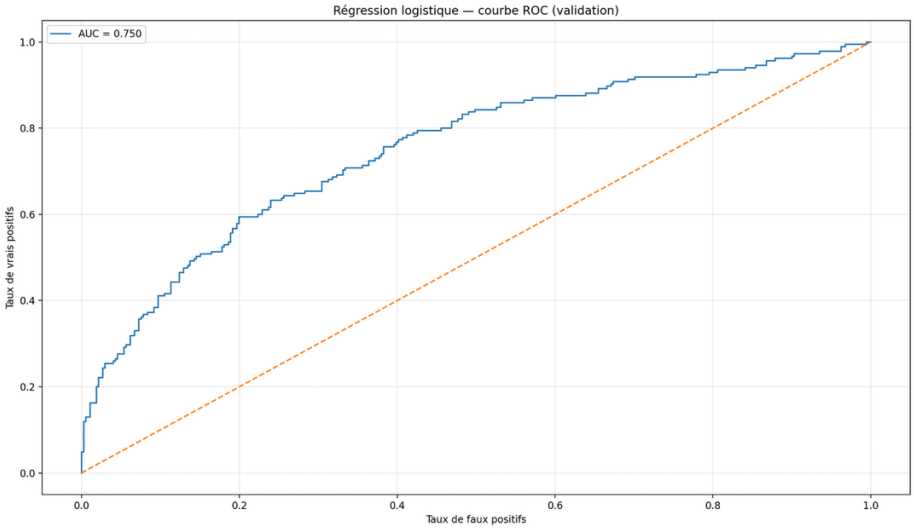
Figure 4.3a – Matrice de confusion de la régression logistique sur la validation.



La matrice de confusion de la régression logistique (Figure 4.3a) confirme son statut de baseline solide. Sur l'échantillon de validation, le modèle identifie 95 épisodes de régime tendu sur 185, soit un rappel d'environ 51%. En parallèle, il ne déclenche un faux signal de stress que pour 66 dates sur 371 où le marché reste en réalité en régime normalisé, ce qui

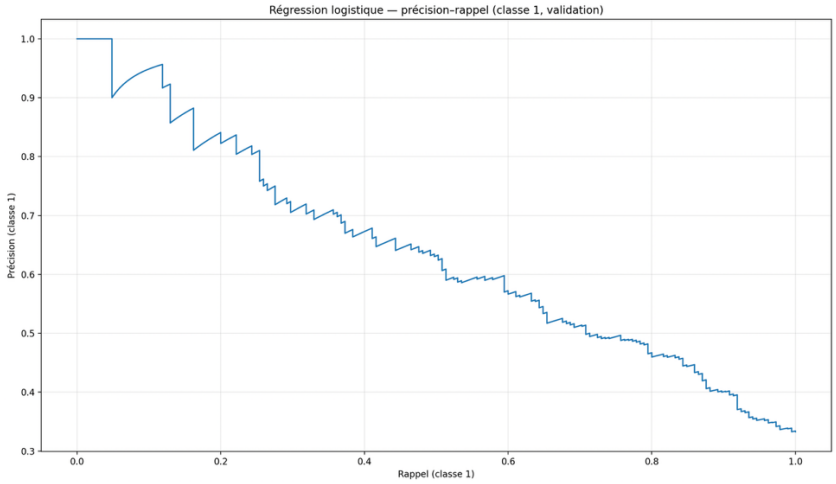
correspond à un taux de faux positifs proche de 18%. Autrement dit, le classificateur parvient à capturer une part significative des épisodes de forte volatilité tout en conservant un niveau de bruit raisonnable sur la classe calme.

Figure 4.3b – Courbe ROC de la régression logistique.



La courbe ROC associée (Figure 4.3b) se situe nettement au-dessus de la diagonale aléatoire. L'aire sous la courbe, $AUC \approx 0,75$, traduit une bonne capacité de ranking probabiliste : en moyenne, lorsqu'on compare deux dates tirées au hasard, l'une en régime calme et l'autre en régime tendu, le modèle attribue une probabilité plus élevée au bon scénario dans trois cas sur quatre. C'est cohérent avec la bonne balanced accuracy observée en 4.1.

Figure 4.3c – Courbe précision–rappel (classe 1) de la régression logistique.

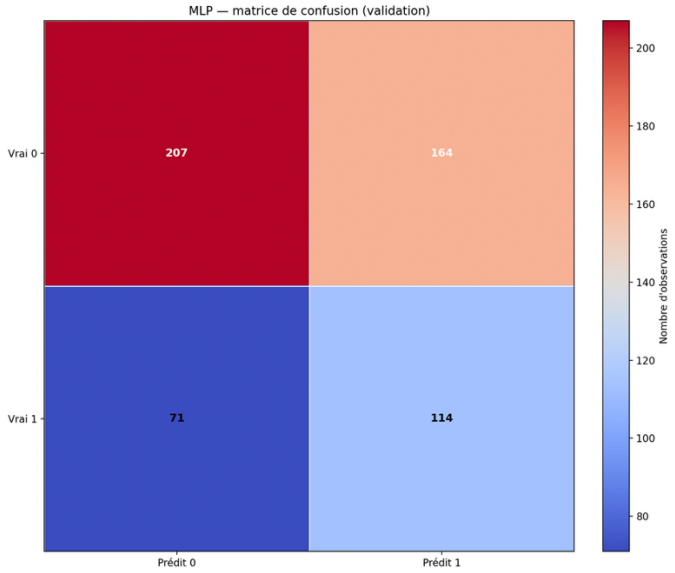


La courbe précision–rappel pour la classe 1 (Figure 4.3c) affine cette lecture. Pour des seuils élevés, le modèle atteint des précisions supérieures à 0,9 au prix d'un rappel plus faible : on détecte alors uniquement les épisodes de stress les plus "évidents". Lorsqu'on diminue le seuil, le rappel augmente mais la précision décroît progressivement vers des valeurs proches de 0,5–0,6. Ce compromis est typique d'un modèle bien calibré sur une classe minoritaire : on dispose d'un curseur explicite permettant d'ajuster la tolérance aux faux signaux selon l'usage (déclenchement d'alertes de risque, filtrage de régimes pour un modèle de pricing, etc.).

Dans l'ensemble, la régression logistique fournit donc un profil de détection équilibré : elle ne capture pas tous les épisodes de forte volatilité, mais produit un ranking stable et contrôlable, ce qui en fait une référence robuste pour la suite

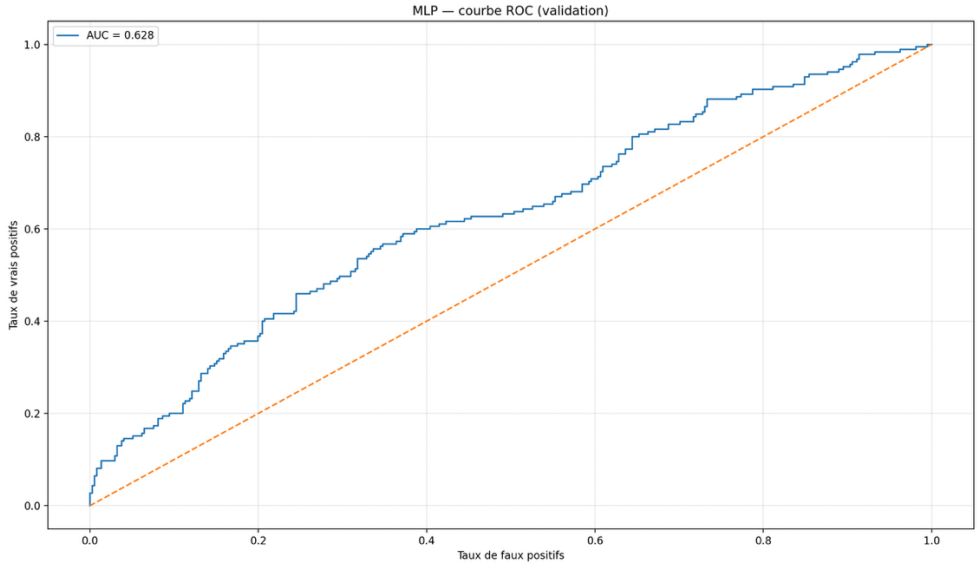
B) Perceptron multicouche

Figure 4.3d – Matrice de confusion du MLP sur la validation.



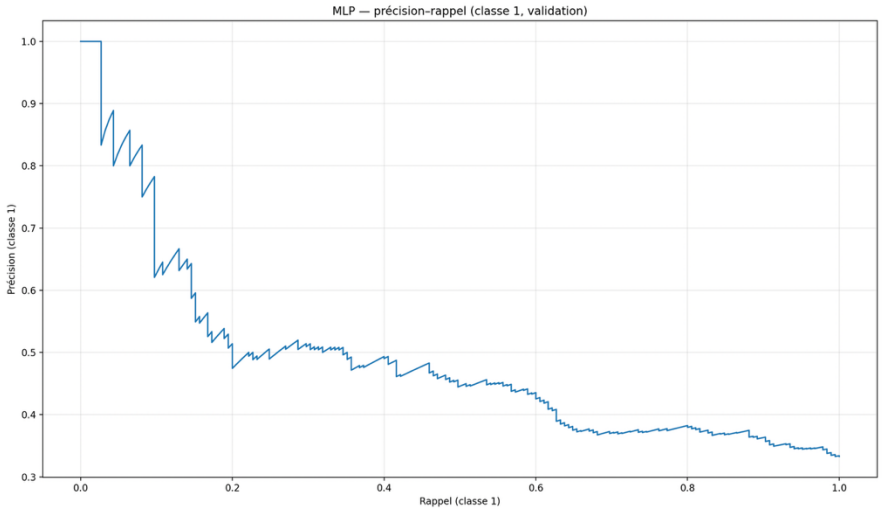
Le perceptron multicouche adopte un comportement sensiblement plus agressif vis-à-vis de la classe tendue. La matrice de confusion (Figure 4.3d) montre que le MLP identifie 114 régimes 1 sur 185, soit un rappel d'environ 62%, supérieur à celui de la régression logistique. Ce gain se paie toutefois par une forte augmentation du bruit : le nombre de faux positifs passe à 164 dates (contre 66 pour la régression logistique), ce qui correspond à un taux de faux signaux proche de 44 % sur la classe calme. En pratique, cela se traduit par une forte hausse du taux de faux positifs : le modèle déclenche fréquemment des signaux de stress alors que le marché reste en régime de volatilité normalisée : il alerte davantage, mais au prix de nombreux épisodes où le marché reste en réalité dans un régime de volatilité normalisée.

Figure 4.3e – Courbe ROC du MLP



La courbe ROC (Figure 4.3e) reflète cette situation intermédiaire. L'AUC se situe autour de 0,63, nettement au-dessus du hasard, mais en deçà de la régression logistique. Le MLP est capable de produire un ranking probabiliste non trivial entre régimes, mais cette hiérarchie est moins nette : pour un même niveau de rappel, le taux de faux positifs reste plus élevé.

Figure 4.3f – Courbe précision–rappel (classe 1) du MLP.



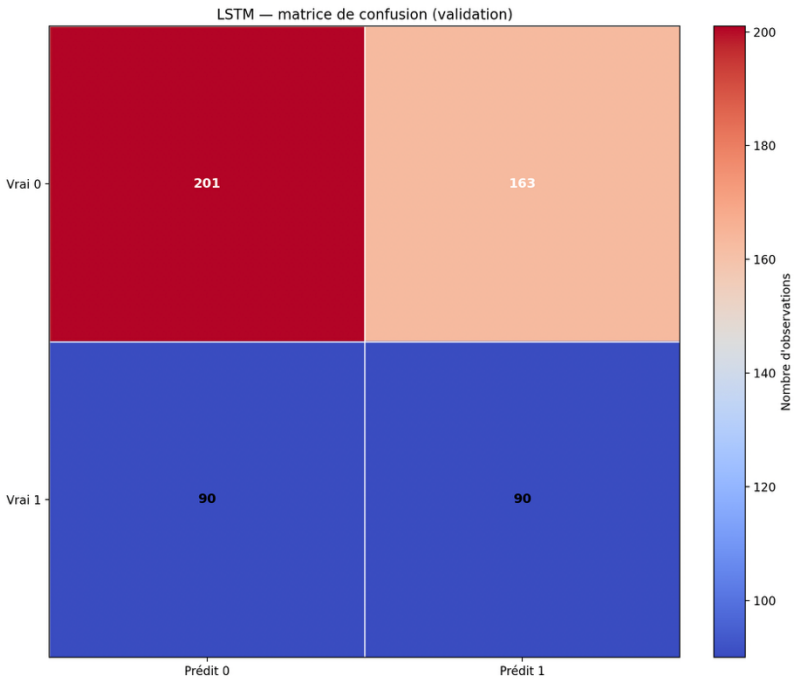
La courbe précision–rappel de la classe 1 (Figure 4.3f) confirme ce diagnostic. Si la précision atteint brièvement des valeurs proches de 1 pour un rappel quasi nul (seuil extrêmement conservateur), elle décroît rapidement vers des niveaux de l'ordre de 0,4–0,5 lorsque l'on cherche à récupérer une fraction significative des régimes tendus. Le gain de

rappel par rapport à la régression logistique se fait donc au prix d'une dégradation marquée de la précision : le MLP fournit une option "plus sensible" pour détecter des stress, mais sans amélioration globale du compromis.

D'un point de vue R&D, ce comportement suggère que la capacité non linéaire du MLP est sous-exploité par rapport à la structure réelle du signal : le modèle parvient à sur-adapter certaines configurations extrêmes, mais ne stabilise pas un ranking probabiliste meilleur que celui de la baseline linéaire.

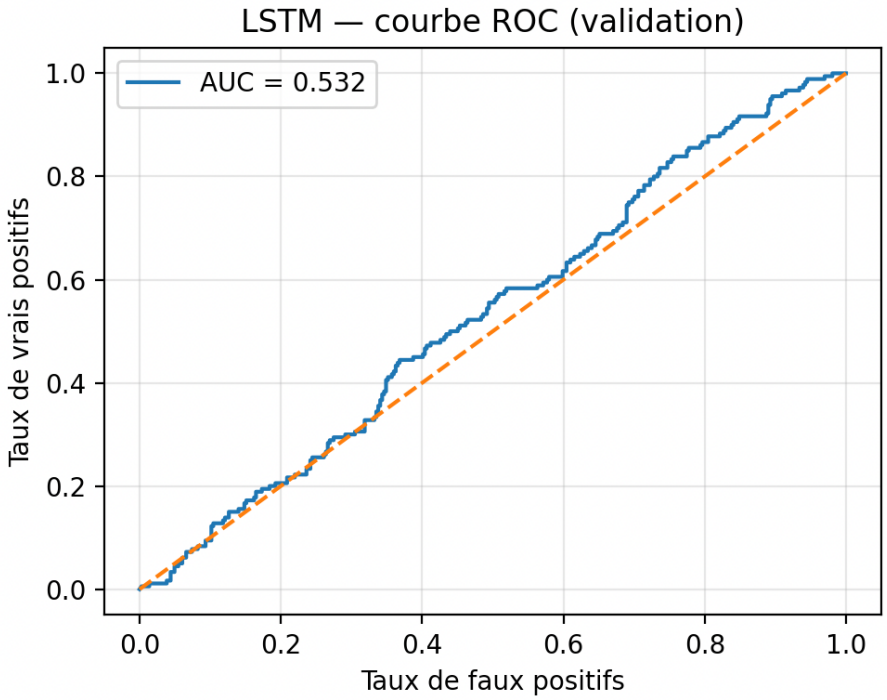
C) LSTM

Figure 4.3g – Matrice de confusion du LSTM sur la validation.



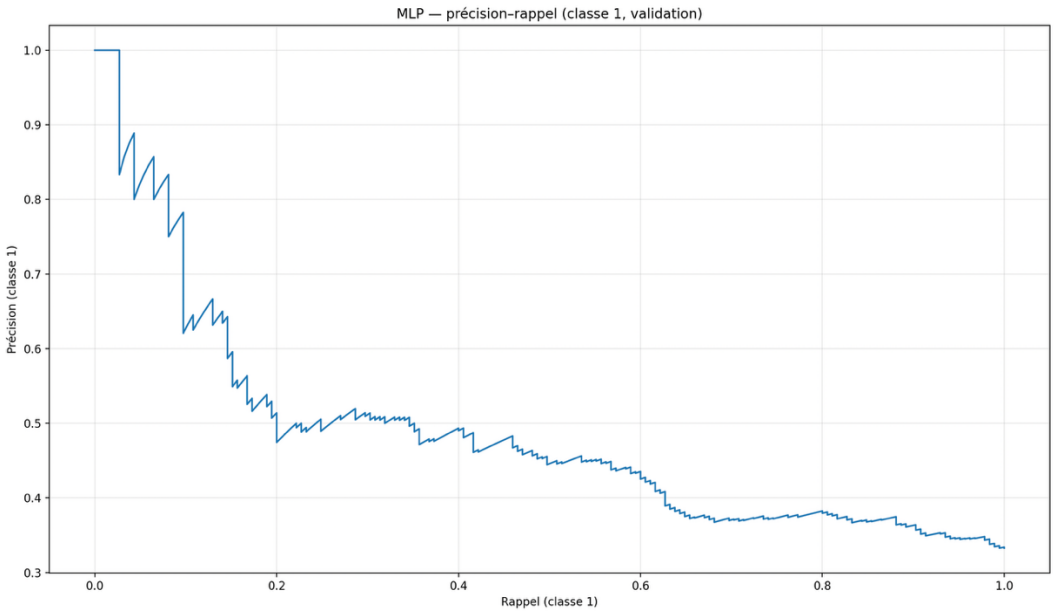
Le LSTM présente un tout autre profil. Sa matrice de confusion (Figure 4.3g) est presque symétrique : le réseau détecte 90 régimes tendus sur 180 environ, soit un rappel voisin de 50%, et génère un nombre comparable de faux positifs sur la classe calme (163 dates). La balanced accuracy se retrouve ainsi légèrement au-dessus de 0,5, exactement dans l'ordre de grandeur observé au Tableau 1. Le modèle n'échoue pas complètement, mais son pouvoir de discrimination net entre régimes reste très limité.

Figure 4.3h – Courbe ROC du LSTM



La courbe ROC du LSTM (Figure 4.3h) se situe à peine au-dessus de la diagonale aléatoire, avec un AUC d'environ 0,53. Cela signifie qu'en moyenne, le réseau n'est guère meilleur qu'un tirage au sort pour ordonner les dates par probabilité de régime tendu. Ce constat fait écho aux courbes d'apprentissage de la section 4.2 : le LSTM mémorise parfaitement l'échantillon d'entraînement, mais la structure temporelle qu'il a captée ne se traduit pas par un ranking robuste hors-échantillon.

Figure 4.3i – Courbe précision–rappel (classe 1) du LSTM.



La courbe précision–rappel (Figure 4.3i) achève de montrer la faiblesse du signal exploité. La précision oscille autour de 0,3–0,4 sur une large plage de rappels, sans plateau clairement dominant. Là où la régression logistique offrait une zone de travail avec une précision nettement supérieure à 0,6, le LSTM propose un compromis presque indifférencié : augmenter le rappel ne dégrade plus tellement la précision, tout simplement parce que l'ensemble du classement est déjà très bruité.

Dans ce cadre expérimental précis (fréquence journalière, label binaire de volatilité future, architecture LSTM relativement simple), la mémoire séquentielle captée par le réseau apparaît donc faiblement exploitable pour distinguer durablement les régimes.

D) Lecture croisée

Pris ensemble, ces diagnostics par régime convergent vers une image cohérente avec la section 4.1 :

- La régression logistique fournit le meilleur équilibre entre détection des régimes tendus, contrôle des faux positifs et qualité du ranking probabiliste.

- Le MLP augmente le rappel sur la classe de stress, mais au prix d'un flot de faux signaux qui dégrade la lisibilité opérationnelle du modèle.

- Le LSTM, malgré des courbes d'apprentissage très flatteuses en entraînement, n'apporte pas de gain structuré : ses matrices de confusion, ses courbes ROC et précision–rappel témoignent d'un signal séquentiel encore trop fragile à cette granularité.

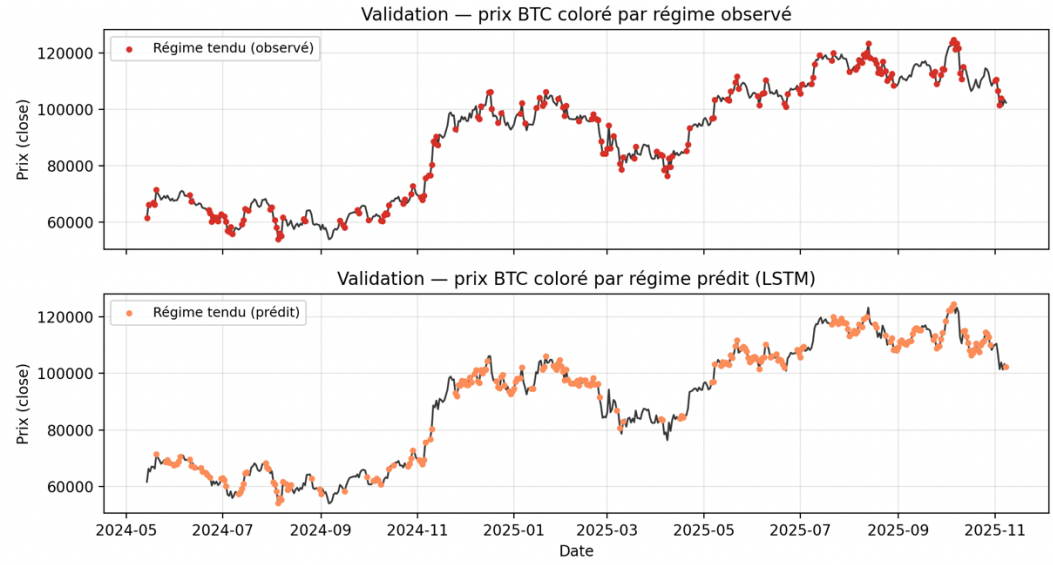
Pour la suite du programme de recherche, ces éléments plaident pour un double mouvement : stabiliser d'abord le signal statique (label, features, calibration) autour de la baseline logistique, puis n'introduire des architectures séquentielles plus sophistiquées qu'en présence d'indices plus clairs de mémoire de régime : par exemple à des fréquences plus fines, sur des labels ajustés ou avec des contraintes de régularisation plus fortes.

4.4 Structure séquentielle et diagnostics graphiques

Les sections précédentes se concentraient sur des métriques agrégées (accuracy, AUC, F1, log-perte). Pour apprécier concrètement ce que "voit" le modèle séquentiel, on examine maintenant la manière dont le LSTM organise les régimes au fil du temps sur le bloc de validation. On se focalise ici sur le LSTM, qui est le seul modèle à disposer d'une mémoire explicite sur 60 jours ; les baselines instantanées ne produiraient qu'un signal point-par-point, sans structure temporelle propre.

La Figure 4.4a superpose le prix quotidien du BTC avec la coloration des régimes observés (panneau supérieur) et prédits par le LSTM (panneau inférieur).

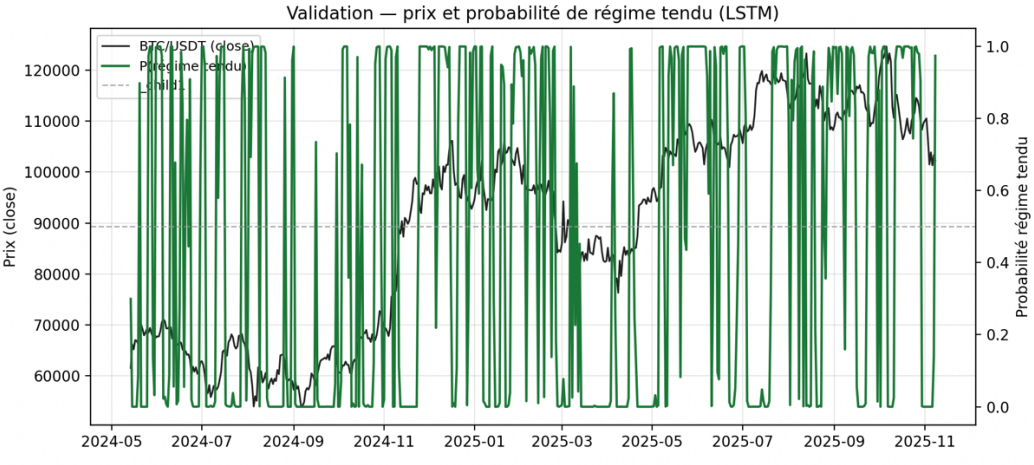
Figure 4.4a : prix BTC coloré par régime observé / prédit



Visuellement, les régimes "tendus" observés se concentrent, comme attendu, autour des phases de volatilité marquée (accélérations haussières, corrections rapides), tandis que les périodes de range ou de tendance plus régulière restent majoritairement en régime "normalisé". Le panneau inférieur montre que le LSTM reprend bien ces grands motifs : les points orange s'agrègent sur les mêmes zones de pente forte et de retournements, ce qui indique que le modèle réagit aux épisodes de stress. En revanche, la densité de signaux tendus prédits est nettement plus élevée que dans les labels observés : le LSTM élargit systématiquement les plages de régime 1 autour des épisodes de volatilité, ce qui se traduit, dans les matrices de confusion, par un taux de faux positifs élevé pour la classe tendue.

La Figure 4.4b détaille la trajectoire probabiliste sous-jacente. Elle représente le prix BTC (axe de gauche) et la probabilité prédite de régime tendu $p_t = \mathbb{P}(y_t = 1 \mid x_{t-59:t})$ (axe de droite), avec le seuil neutre 0,5 matérialisé par une ligne en pointillés.

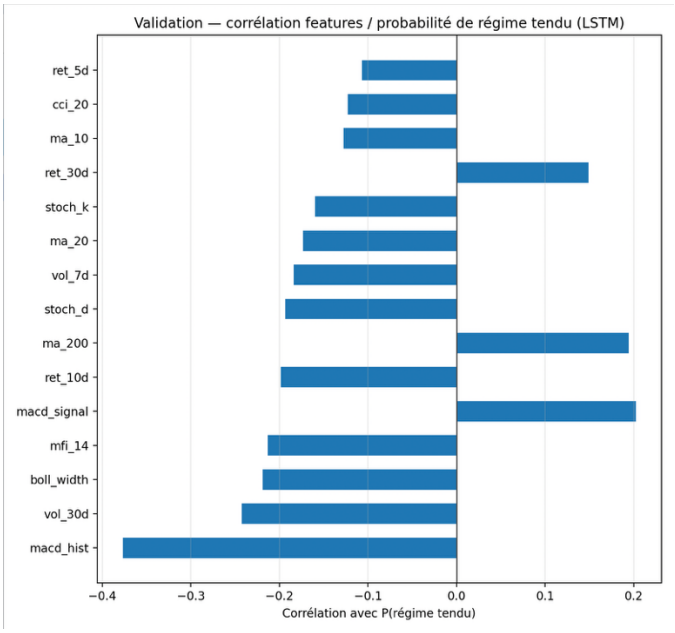
Figure 4.4b : prix et probabilité de régime tendu



La probabilité de régime tendu adopte un profil quasi binaire : elle alterne rapidement entre des niveaux proches de 0 et proches de 1, avec très peu de valeurs intermédiaires stables. Ce comportement est cohérent avec les diagnostics de log-perte : le LSTM devient extrêmement confiant sur ses prédictions, mais cette confiance ne généralise pas bien hors-échantillon, d'où une log-perte de validation élevée. On retrouve également le biais déjà mis en évidence : sur de longues portions de l'échantillon p_t reste très souvent au-dessus de 0,5, ce qui ancre le modèle dans un régime "stress" quasi permanent. Le signal séquentiel capturé par le LSTM est donc réel (les pics de probabilité coïncident globalement avec les épisodes de volatilité), mais il est exploité de façon trop agressive pour produire un classement probabiliste robuste.

Enfin, la Figure 4.4c cherche à relier ce signal de régime aux variables explicatives sous-jacentes. Elle montre, sur le bloc de validation, les corrélations de Pearson entre la probabilité de régime tendu p_t et un sous-ensemble de features clefs, triées par valeur absolue.

Figure 4.4c : corrélation features / probabilité de régime tendu



Plusieurs motifs apparaissent clairement. D'une part, des indicateurs de momentum et de "trend strength" comme macd_hist, macd_signal, ret_10d ou les moyennes mobiles (ma_20, ma_200) présentent des corrélations marquées avec p_t : les phases de momentum haussier soutenu (histogramme MACD largement positif, prix au-dessus de la moyenne long terme) sont associées à des probabilités de stress plus faibles, tandis que les configurations de momentum dégradé ou de retournement s'accompagnent de p_t élevés. D'autre part, les mesures de volatilité et de largeur de bande (vol_30d, vol_7d, boll_width) sont, comme attendu, positivement corrélées à la probabilité de régime tendu : lorsque la dispersion des rendements et l'amplitude des bandes de Bollinger augmentent, le LSTM tend à basculer en régime 1. Enfin, des indicateurs de flux et de liquidité comme mfi_14 contribuent également au signal, en modulant la probabilité de stress en fonction de la pression acheteuse ou vendeuse.

Ces diagnostics graphiques complètent ainsi les tableaux de performance : ils montrent que, malgré un sur-apprentissage prononcé et une calibration probabiliste perfectible, le LSTM exploite bien des patterns séquentiels économiquement plausibles – alternance de phases de calme et de stress liées à la combinaison de momentum, de volatilité et de structure de tendance. La question n'est donc pas tant de savoir si une "mémoire de marché" existe, mais de la contraindre et de la régulariser suffisamment pour obtenir un signal de régime exploitable à fréquence quotidienne.

4.5 Résultats des ablations

Les expériences précédentes se concentraient sur un LSTM « complet » observant 60 jours d'historique et exploitant l'ensemble des indicateurs décrits en section 2. Cette sous-section examine, à architecture fixe, l'impact (i) de la suppression explicite des features de volatilité et (ii) du raccourcissement de la fenêtre séquentielle à 15 jours. L'objectif n'est pas d'optimiser la performance absolue – qui reste inférieure à celle de la régression logistique – mais de documenter la sensibilité du signal séquentiel aux briques d'information les plus naturelles : taille de la mémoire et mesure de la dispersion des rendements.

Les trois variantes considérées sont :

- full_60d : LSTM avec toutes les features, séquence de 60 jours ;
- no_vol_60d : mêmes features, à l'exception de vol_7d, vol_30d et boll_width, toujours sur 60 jours ;
- full_15d : LSTM complet mais avec une fenêtre raccourcie à 15 jours.

Les métriques de validation correspondantes sont résumées dans le Tableau 2.

Variante	Accuracy	Balanced accuracy	Recall (classe 1)	F1 (classe 1)	Log-perte	AUC
full_60d	0,535	0,518	0,467	0,399	2,38	0,53
no_vol_60d	0,586	0,544	0,417	0,400	1,73	0,60
full_15d	0,568	0,569	0,574	0,468	2,29	0,58

Deux enseignements principaux se dégagent. Premièrement, les features de volatilité ne constituent pas, dans ce protocole, un levier évident de généralisation pour le LSTM. La variante no_vol_60d améliore l'accuracy (≈ 0,59) et l'AUC (≈ 0,60) par rapport au modèle complet sur 60 jours, tout en maintenant un F1 pour la classe tendue du même ordre de grandeur. La baisse du rappel sur les régimes tendus (≈ 0,42 versus ≈ 0,47) indique que le LSTM sans volatilité devient légèrement plus conservateur sur la détection des épisodes de stress, ce qui est cohérent avec la réduction d'information sur l'ampleur des mouvements. Le fait que cette ablation n'entraîne pas de dégradation catastrophique suggère que la structure du régime est déjà largement intégrée via les indicateurs de tendance, de momentum et de structure de prix.

Deuxièmement, la profondeur de mémoire joue un rôle plus ambigu. Le modèle full_15d – qui n'observe que 15 jours d'historique – atteint la meilleure balanced accuracy (≈ 0,57) et le meilleur rappel de la classe tendue (≈ 0,57), au prix d'un AUC intermédiaire (≈ 0,58) et d'une log-perte encore élevée. Autrement dit, un horizon plus court semble aider le LSTM à mieux équilibrer les deux régimes en fréquence, et à capter davantage d'épisodes de forte volatilité, mais sans pour autant produire un ranking probabiliste plus robuste. Cette sensibilité à la longueur de séquence est cohérente avec la structure des clusters de volatilité observés empiriquement (section 3.3) : une fenêtre très longue (60 jours) mélange potentiellement plusieurs micro-régimes et renforce le risque d'overfitting séquentiel, tandis qu'une fenêtre plus courte réagit davantage aux configurations locales.

Dans tous les cas, aucune de ces variantes n'égale la régression logistique en termes d'AUC ou de log-perte. Les ablations doivent donc être lues comme des diagnostics internes : elles montrent que le comportement du LSTM est effectivement modulé par la présence explicite de la volatilité et par la profondeur de mémoire, mais qu'à cette granularité journalière la "mémoire de marché" capturable par un LSTM vanilla reste faible, et nécessite des architectures plus contraintes ou des labels plus ciblés pour devenir exploitable.

4.6 Illustration d'usage : stratégie de filtrage de risque

Pour illustrer de façon concrète ce que l'on peut faire avec le signal de régime $\hat{p}_t = \mathbb{P}(Y_t = 1 \mid \mathcal{F}_t)$ issu du LSTM, on construit une stratégie jouet de filtrage de risque sur le bloc de validation. L'idée est volontairement minimale : lorsque la probabilité de régime tendu dépasse un certain seuil, l'allocation se replie en cash ; sinon, elle reste entièrement investie en BTC.

Soit P_t le prix de clôture et

$$r_t = \frac{P_t}{P_{t-1}} - 1$$

le rendement simple quotidien. On définit une exposition binaire

$$e_t(\tau) = \begin{cases} 1 & \text{si } \hat{p}_t < \tau, \\ [4pt]0 & \text{sinon}, \end{cases}$$

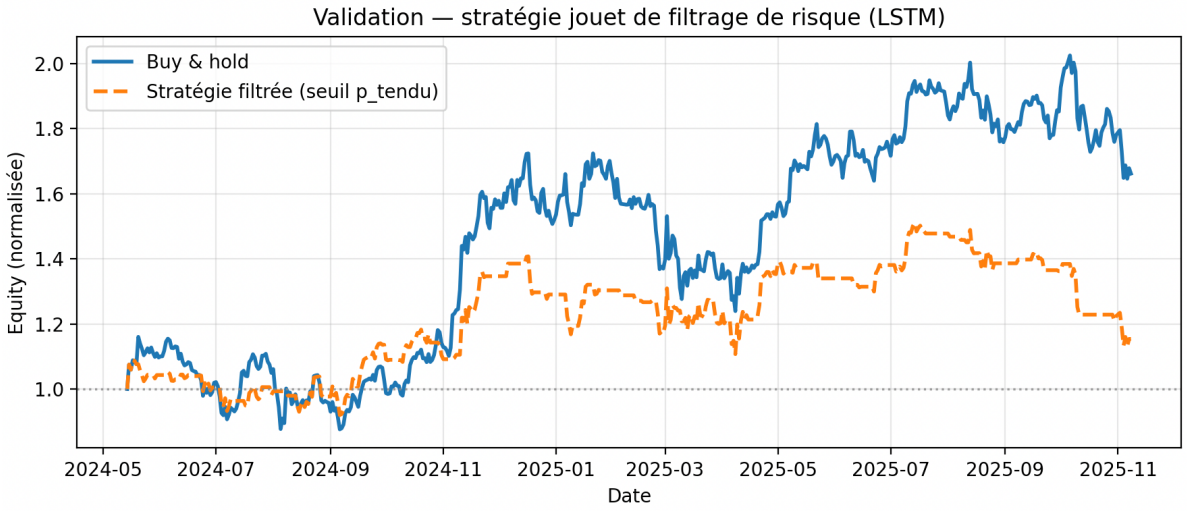
avec un seuil neutre $\tau = 0,5$, cohérent avec la règle de décision binaire utilisée dans les sections précédentes. La trajectoire d'equity de la stratégie filtrée s'écrit alors

$$E_t^{\text{filt}}(\tau) = E_0 \prod_{u=1}^{t} (1 + e_u(\tau)\, r_u),$$

tandis que le buy–and–hold passif correspond au cas $e_u \equiv 1$,

$$E_t^{\text{BH}} = E_0 \prod_{u=1}^{t} (1 + r_u).$$

Figure 4.6 : Toy Vs Buy and hold



La Figure 4.6 montre l'equity normalisée de ces deux stratégies sur le bloc de validation. Visuellement, la courbe filtrée suit le profil général du marché, mais avec des phases de plateau lorsque le modèle anticipe un régime tendu et coupe l'exposition.

Pour quantifier cette illustration, on calcule sur les rendements quotidiens correspondants :

- le rendement cumulé $\text{cum\_return} = E_T/E_0 - 1$;
- la volatilité annualisée $\text{vol\_annual} = \sqrt{365}\ \sigma(r_t)$ (fréquence quotidienne) ;
- le maximum drawdown, $\min_t \left(E_t / \max_{u \le t} E_u - 1\right)$, sans taux sans risque.

Les résultats sont récapitulés dans le Tableau 3.

Tableau 3 – Statistiques de la stratégie jouet de filtrage de risque (bloc de validation)

Stratégie	cum_return	vol_annual	max_drawdown	Sharpe
Buy & hold	0,6615	0,4545	-0,2810	0,9749
Stratégie filtrée LSTM	0,1539	0,3387	-0,2506	0,4517

Sur cette fenêtre, la stratégie filtrée réduit effectivement la volatilité et le drawdown par rapport au buy–and–hold, mais au prix d'un rendement cumulé nettement inférieur et d'un Sharpe divisé par plus de deux. Autrement dit, le signal de régime fourni par le LSTM permet déjà de produire une trajectoire d'equity plus lissée, mais il n'est pas, en l'état, suffisamment propre ni bien calibré pour améliorer le couple rendement/risque de manière robuste.

Cet exercice doit donc être lu comme une illustration méthodologique, et non comme un backtest opérationnel : le seuil τ n'est pas optimisé, les coûts de transaction sont négligés, et aucune contrainte d'implémentation réelle n'est prise en compte. Il montre néanmoins comment un signal de régime $\hat{p}_t$ peut être encapsulé dans une règle d'allocation simple de type « long/cash », et préfigure l'usage cible de ce type de modèle : servir de brique de filtrage du risque dans des stratégies d'allocation plus complètes (multi-actifs, multi-horizons), où le signal de régime viendrait moduler l'exposition globale plutôt que générer seul des décisions de trading.

V Discussion et limites

5.1 Lecture critique des résultats empiriques

Les résultats empiriques obtenus dans cette première itération confirment qu'une partie significative de la dynamique de régime sur Bitcoin peut déjà être capturée par des modèles statiques relativement simples. À protocole expérimental constant même vecteur de features, même définition du label de régime, même découpage temporel la régression logistique s'impose comme référence de base. La frontière linéaire qu'elle apprend, appliquée à l'instantané des indicateurs techniques, parvient à séparer de manière robuste les phases de volatilité future "normalisée" des épisodes plus tendus, avec une combinaison équilibrée d'accuracy globale, de balanced accuracy et d'AUC. Autrement dit, la simple structure affine dans l'espace des features organise déjà le marché en deux régimes économiquement lisibles.

Le perceptron multicouche n'apporte qu'un bénéfice marginal. La non-linéarité supplémentaire permet certes de rehausser ponctuellement le rappel sur la classe de stress, mais au prix d'une dégradation de la calibration probabiliste et d'une augmentation sensible des faux signaux. Le compromis global reste moins favorable que celui de la régression

logistique, ce qui suggère que, dans ce cadre précis, le gain d'expressivité ne compense pas le coût de variance ajoutée par l'architecture.

Le LSTM, conçu pour exploiter explicitement 60 jours d'historique, met en évidence un phénomène différent. Sur l'échantillon d'entraînement, le modèle converge vers une quasi-reconstruction parfaite des labels, avec une log-perte très faible et des probabilités extrêmes proches de 0 ou 1. Sur le bloc de validation, cette confiance se retourne en fragilité : l'accuracy et la balanced accuracy se rapprochent d'un niveau à peine supérieur au hasard, l'AUC recule nettement et la log-perte augmente de manière significative. Les diagnostics graphiques confirment ce diagnostic : la probabilité de régime tendu $\hat{p}_t$ adopte un profil quasi binaire, basculant violemment entre les deux extrêmes, et reste durablement ancrée au-dessus du seuil 0,5 sur de longues périodes. Les épisodes de volatilité marquée sont bien détectés, mais le modèle tend ensuite à étendre exagérément la zone de stress, ce qui se traduit mécaniquement par un excès de faux positifs dans les matrices de confusion.

Les expériences d'ablation réalisées à architecture constante apportent un éclairage complémentaire. La suppression des features directement liées à la volatilité ne provoque pas d'effondrement du LSTM ; au contraire, certaines métriques hors-échantillon s'améliorent légèrement, en particulier l'accuracy et l'AUC, même si le signal de stress devient un peu plus conservateur. Cela laisse penser que la structure de régime est déjà largement encodée dans les indicateurs de tendance, de momentum et de structure de prix, et que les proxies de volatilité, dans ce paramétrage initial, n'apportent qu'un gain d'information marginal. De même, le raccourcissement de la séquence à quinze jours améliore la balanced accuracy et le rappel de la classe tendue, sans pour autant rattraper la régression logistique en termes de qualité probabiliste. Une fenêtre plus courte semble donc permettre au LSTM de mieux suivre les configurations locales de marché, mais sans lui donner encore un véritable avantage structurel sur les baselines statiques.

Enfin, la stratégie de filtrage de risque construite à partir de $\hat{p}_t$ joue pleinement son rôle d'illustration. Sur le segment de validation, la trajectoire d'equity filtrée, qui réduit l'exposition dès que la probabilité de régime tendu dépasse un seuil neutre, présente une volatilité et un drawdown maximal plus contenus que le buy-and-hold, mais au prix d'un rendement cumulé nettement inférieur. Dans la configuration actuelle, le signal de régime agit donc davantage comme un dispositif de freinage capable de couper une partie des extrémités de distribution que comme un moteur d'amélioration du couple rendement/risque. Cet exercice doit être lu comme une preuve de concept : il montre que le signal peut être injecté dans une logique de contrôle du risque, mais ne constitue pas, en l'état, un overlay prêt pour une mise en production.

5.2 Portée et limites du cadre expérimental

Le cadre expérimental retenu dans cette première note est volontairement restrictif et explique en partie la hiérarchie observée entre modèles. La granularité temporelle, d'abord, est exclusivement journalière. Le label de régime est défini à partir de la volatilité réalisée future sur un horizon court, mais toute la micro-dynamique intraday – là où se cristallisent souvent les transitions de régime, les séquences de liquidations et les chocs de

liquidité – est, par construction, absente. Dans ce contexte, l'avantage informationnel d'un modèle séquentiel est mécaniquement limité : la mémoire qu'il peut exploiter n'est qu'une mémoire d'"états journaliers déjà agrégés".

L'univers d'actifs est, ensuite, réduit au seul BTC spot. Si Bitcoin constitue un laboratoire naturel pour ce type d'exercice, il ne représente ni la diversité des profils de risque des autres cryptoactifs, ni les interactions de régimes entre actifs liés (ETH, indices de marché, produits dérivés). Il n'est donc pas possible, à ce stade, de conclure sur la transférabilité du signal de régime à un univers multi-actifs, ni sur son comportement dans des portefeuilles réellement diversifiés.

Le vecteur de features repose essentiellement sur des constructions issues des prix et des volumes : rendements multi-horizons, indicateurs de momentum, mesures de tendance, volatilités réalisées et largeurs de bandes, proxies de participation via volumes relatifs et MFI. Les dimensions on-chain, dérivées (basis futures, funding, skew d'options) ou macro-financières ne sont pas encore intégrées ; de même, aucune information de carnet d'ordres ou de microstructure n'est utilisée. Le modèle apprend donc un régime au sens strict de "régime de prix", sans vision directe de la structure de flux ou des contraintes de financement qui peuvent, en pratique, déclencher ou prolonger des phases de stress.

Le choix du label de régime constitue une autre simplification importante. La dichotomie "calme / tendu" dérivée d'un ratio de volatilité future sur une volatilité de référence offre une grille de lecture claire, mais unidimensionnelle. Elle ne distingue pas les épisodes de stress haussier des chocs baissiers, ne tient pas compte de la profondeur des drawdowns, ni des propriétés de récupération post-crise. Plusieurs configurations de marché très différentes peuvent être agrégées dans la même classe de régime, ce qui limite mécaniquement la quantité d'information exploitable par les modèles.

Enfin, l'architecture séquentielle elle-même reste minimaliste. Le LSTM utilisé est volontairement simple, avec une seule couche, un nombre d'unités maîtrisé, et sans recours à des mécanismes de régularisation avancés (dropout, pénalisation spécifique des poids récurrents, calibrations ex-post des probabilités). L'évaluation hors-échantillon repose sur un unique split chronologique et ne met pas encore en œuvre de schémas de validation croisée temporelle plus sophistiqués. Quant à la stratégie de filtrage de risque, elle est volontairement nette exposition binaire, absence de frais de transaction, absence de contraintes de turnover afin de rester lisible. L'ensemble de ces choix sont adaptés à une première exploration, mais constituent autant de points de vigilance lorsqu'il s'agit d'interpréter la portée des résultats.

5.3 Pistes de développement et intégration produit

Dans cette perspective, les résultats présentés ici doivent être lus comme un jalon de recherche plutôt que comme un aboutissement. Ils confirment d'abord que, à fréquence quotidienne et pour un label de volatilité simple, la majeure partie du signal exploitable sur les régimes de marché est déjà accessible à des modèles statiques bien spécifiés. Ils valident ensuite l'existence d'une mémoire séquentielle non triviale les diagnostics graphiques du LSTM montrent clairement que certaines configurations de trajectoires sont associées à des

profils de risque différenciés mais soulignent qu'une architecture vanilla, peu régularisée, n'est pas suffisante pour transformer cette mémoire en avantage probabiliste robuste.

Les prochains développements s'orienteront donc dans deux directions complémentaires. Sur le plan méthodologique, l'enjeu est d'enrichir simultanément le label de régime, le vecteur de features et les architectures séquentielles, de manière à mieux aligner la définition du problème de prédiction avec les besoins d'un moteur d'allocation réel. Cela implique d'explorer des labels multi-niveaux (par exemple en intégrant explicitement la dimension de drawdown ou des horizons de risque différenciés), d'ouvrir le jeu de données à des signaux on-chain, dérivés et macro, et de tester des modèles plus structurés architectures convolutives temporelles, mécanismes d'attention, modèles à changement de régime explicite avec des protocoles de régularisation et de validation croisée adaptés à la faible fréquence des observations.

Sur le plan "produit", l'objectif est de transformer le signal de régime en véritable brique de gestion. La stratégie jouet de filtrage présentée en section 4.6 fournit un canevas naturel pour développer des overlays plus réalistes : expositions graduelles plutôt que binaires, intégration de coûts de transaction, cibles de volatilité explicites, contraintes de drawdown, interaction avec d'autres moteurs d'alpha directionnels ou relatifs. À terme, l'ambition est que ce type de modèle de régime ne soit plus uniquement un objet de recherche isolé, mais qu'il alimente un bloc de contrôle du risque au sein d'une architecture plus large d'allocation systématique, où chaque moteur : tendance, carry, signaux on-chain, facteurs macro dialogue avec une estimation cohérente de l'état de marché.

Dans cette optique, la présente note joue pleinement son rôle : elle établit une baseline quantitative robuste, met en évidence les limites des approches séquentielles naïves et trace un chemin clair vers les itérations suivantes du programme de R&D.

Conclusion

Cette première série d'expérimentations séquentielles sur le Bitcoin avait un objectif volontairement circonscrit : tester, sur un cadre simple mais propre, l'existence d'une mémoire exploitable dans l'enchaînement des régimes de volatilité. À partir d'un label binaire fondé sur la volatilité réalisée future, d'un jeu compact d'indicateurs techniques normalisés et d'un protocole d'entraînement strictement causal, trois familles de modèles ont été mises en regard : une régression logistique, un MLP sans mémoire et un LSTM observant 60 jours d'historique. Les résultats empiriques sont sans ambiguïté : dans ce cadre précis, la baseline linéaire surperforme systématiquement les architectures plus expressives en termes d'AUC et de log-perte, tandis que le LSTM sur-apprend rapidement et ne parvient pas à transformer son accès à l'historique en gain robuste hors-échantillon.

Ce constat ne remet pas en cause l'hypothèse d'une organisation du marché en régimes, mais en borne la portée à cette granularité. À fréquence journalière, une part significative de l'information de régime semble déjà contenue dans la configuration instantanée des indicateurs (rendements multi-horizons, volatilité réalisée, écarts aux

moyennes mobiles, proxies de participation) ; la composante proprement séquentielle, telle qu'un LSTM vanilla peut l'exploiter sur 15 à 60 jours, apparaît de faible amplitude et très sensible au sur-apprentissage. Les ablations menées sur la longueur de séquence et sur les features de volatilité confirment ce diagnostic : le comportement du LSTM est bien modulé par ces briques d'information, mais aucun réglage simple ne permet de franchir de manière stable le plafond fixé par la régression logistique.

Pour autant, l'exercice n'est pas négatif ; il est structurant. La note a permis (i) de poser un pipeline de données et de features entièrement reproductible, (ii) de définir un label de régime explicite, ancré dans la volatilité réalisée future plutôt que dans des heuristiques ad hoc, (iii) de comparer de façon homogène des modèles avec et sans mémoire, et (iv) de documenter la structure séquentielle du signal de régime via des diagnostics graphiques et une stratégie jouet de filtrage de risque. L'ensemble fournit une base empirique claire : il existe bien un signal de régime exploitable sur BTC au quotidien, mais ce signal est essentiellement "statique" dans ce cadre, et ne justifie pas encore l'usage d'architectures séquentielles complexes en production.

Les limites du dispositif sont, en miroir, tout aussi claires : un seul actif (BTC), une fréquence daily, un label binaire de volatilité future relativement simple, une architecture LSTM volontairement minimale et l'absence de backtests pleinement opérationnels intégrant frictions et contraintes d'allocation. Ces choix étaient assumés pour cette V1, afin de privilégier la lisibilité du protocole et la traçabilité des résultats ; ils fixent désormais la feuille de route des itérations suivantes.

La suite du programme de recherche s'organise donc naturellement autour de trois axes. Sur le plan des données et des labels, il s'agira d'explorer des définitions plus riches de régime (multi-classe, vol cible, drawdown anticipé, régimes conjoints prix/volume), de tester des fréquences plus fines (intra-day) et d'étendre l'analyse à un panier d'actifs liquides. Sur le plan des modèles, les efforts porteront sur des architectures séquentielles plus contraintes et mieux régularisées, calibrées en priorité à partir de la baseline logistique. Enfin, sur le plan "produit", le signal de régime sera progressivement intégré, non comme moteur unique de décision, mais comme couche de filtrage de risque et de contrôle d'exposition dans des moteurs d'allocation plus complets.

En ce sens, cette note joue bien son rôle : non pas proposer un modèle séquentiel "clé en main" pour le trading sur Bitcoin, mais clarifier ce que la donnée raconte réellement sur la mémoire de marché à cette échelle, et tracer un chemin réaliste vers des briques de régimes utilisables dans les futurs moteurs d'allocation de HilmarCorp.