Prédiction des Pics de Volatilité

Modélisation avancée sur données financières

Vicente Campello

Avril 2025

Table des matières

1 Introduction

Ce projet vise à anticiper les pics de volatilité sur les marchés cryptos, en particulier sur l'ETH et le BTC. L'objectif est de concevoir un système d'alerte précoce basé sur des modèles LSTM, capables d'extraire des patterns temporels complexes dans les séries financières.

2 Objectif du projet

Ce projet s'inscrit dans une démarche double : à la fois théorique et pratique.

Sur le plan théorique, il vise à mieux comprendre la dynamique de la volatilité financière : sa formation, ses ruptures et ses régularités cachées. Comprendre les cycles de tension sur les marchés est une compétence fondamentale pour quiconque souhaite naviguer intelligemment dans les environnements volatils. Il ne s'agit pas simplement de constater l'instabilité, mais d'en modéliser l'apparition, avec une granularité suffisante pour en tirer des signaux exploitables.

Sur le plan pratique, le but est d'intégrer cette compréhension dans une approche de trading plus rigoureuse. Grâce au modèle développé, il devient possible d'être alerté lorsqu'un pic de volatilité est en cours de formation, et ainsi d'adapter en amont son comportement de marché. Cela permet une meilleure gestion du risque, une prudence accrue dans les zones à haute turbulence, et potentiellement une stratégie plus fine dans l'exécution.

— Comprendre la formation de la volatilité pour en faire un signal exploitable
— Détecter les phases d'instabilité à venir sur les marchés
— Renforcer la vigilance du trader si un spike est anticipé
— Évaluer la faisabilité d'un modèle prédictif basé sur des données publiques
— Passer de l'analyse offline à une prédiction en temps réel

3 Première approche - Analyse technique classique

La première idée a été d'utiliser une approche technique simple pour détecter les pics de volatilé du BTC, à l'aide d'indicateurs tels que la volatilite glissante et des seuils statiques. Voici un extrait de code du fichier `analyse_btc.ipynb` :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("BTC-USD.csv")
df['log_ret'] = np.log(df['Close'] / df['Close'].shift(1))
df['volatility'] = df['log_ret'].rolling(30).std() * np.sqrt(365)

threshold = df['volatility'].quantile(0.99)
df['spike'] = (df['volatility'] > threshold).astype(int)

plt.figure(figsize=(12, 5))
df['Close'].plot(label='BTC Close')
df[df['spike'] == 1]['Close'].plot(style='ro', label='Spike')
```

```
plt.legend()
plt.title("Detection de pics de volatilité par quantile statique")
plt.show()
```

Cependant, cette méthode s'est rapidement révélée limitée : les pics ainsi identifiés étaient souvent soit trop tardifs, soit pas assez représentatifs de réelles tensions de marché. D'où le basculement vers une approche par apprentissage profond.

4 Données et traitement

4.1 Sources

— Historique ETH-USD et BTC-USD (Yahoo Finance, Binance)
— Données journalières sur 10 ans

4.2 Prétraitements

La partie prétraitement constitue le cœur conceptuel du projet. Chaque indicateur a été choisi pour capter une facette différente de la dynamique de marché. L'objectif n'était pas simplement d'entrer des données dans un réseau neuronal, mais de fournir un environnement mathématiquement riche, pertinent, et interprétable pour une prédiction robuste.

— **Calcul du log-return** : Le rendement logarithmique permet de lisser les effets multiplicatifs du prix tout en respectant l'additivité temporelle. Il est à la base de toute modélisation financière sérieuse.
— **Volatilité glissante (rolling std)** : Elle capte la variation locale du prix sur une fenêtre mobile. C'est une première mesure naïve mais efficace de la volatilité historique.
— **Volatilité conditionnelle (GARCH)** : La vraie nature de la volatilité financière est hétéroscédastique, c'est-à-dire qu'elle varie dans le temps avec mémoire. Le GARCH (1,1) permet de modéliser cette mémoire et de projeter une volatilité plus « réaliste » et conditionnée par l'histoire récente du marché.
— **Exposant de Hurst** : Il mesure la régularité (ou l'irrégularité) d'une série. Un H proche de 0.5 indique un comportement aléatoire pur ; en dessous, on observe une anti-persistence ; au-dessus, une tendance. Cette information est cruciale pour déterminer si un pic de volatilité est « isolé » ou s'inscrit dans une structure persistante.
— **Normalisation avec `RobustScaler`** : Le marché étant plein de valeurs extrêmes et de données bruitées, une normalisation robuste (basée sur la médiane et les quantiles) permet de limiter l'impact des outliers sans dénaturer la structure.
— **Ciblage dynamique des spikes** : Plutôt que de définir les spikes par un seuil absolu ou arbitraire, j'ai choisi une approche basée sur une moyenne glissante + 2 écarts-types (rolling 20 jours). Cela permet d'adapter dynamiquement le seuil aux conditions de marché et d'éviter une classification biaisée.

Ce pipeline de prétraitement est donc à la fois mathématiquement rigoureux, économiquement justifié, et conçu pour s'insérer dans une logique de modélisation séquentielle.

5 Modèle et architecture

Le choix de l'architecture du modèle n'a pas été arbitraire. Il repose sur une double intuition : d'une part, que la volatilité suit une logique temporelle complexe, avec des signaux faibles et non linéaires ; d'autre part, que ces signaux sont enfouis dans des patterns locaux parfois très courts, parfois très persistants.

- **Modèle séquentiel : Conv1D + BiLSTM**
  Le coeur du modèle repose sur une architecture hybride combinant deux types de couches :
  - *Conv1D* : ces couches convolutives permettent de capter des motifs locaux sur les séquences temporelles (par exemple, une hausse rapide de la volatilité ou un enchaînement de hausses). Elles jouent un rôle de "filtrage" initial.
  - *BiLSTM* : les réseaux LSTM sont conçus pour apprendre des dépendances temporelles longues. Leur version bidirectionnelle permet de capter des dynamiques à la fois antérieures et postérieures dans la séquence. Cela rend le modèle particulièrement apte à reconnaître les configurations de marché annonciatrices de turbulence.
- **Fenêtre de contexte : 90 jours**
  J'ai opté pour une séquence de 90 jours, ce qui permet au modèle d'intégrer à la fois des événements courts (type panic sell) et des configurations structurelles plus longues (accumulation, compression, etc.). Ce choix reflète une vision swing-trading du marché : on cherche à anticiper les zones de stress avant qu'elles ne se déclenchent.
- **Fonction de perte : BinaryCrossentropy avec `class_weight`**
  Les pics de volatilité étant rares (classe minoritaire), un entraînement naïf aurait poussé le modèle à prédire systématiquement « pas de spike ». Pour pallier ce déséquilibre, j'ai introduit une pondération des classes dans la fonction de perte. Ainsi, les erreurs sur les spikes coûtent davantage que les erreurs sur le calme. Cela force le modèle à rester attentif aux signaux faibles.
- **Sauvegarde du modèle**
  Le modèle est sauvegardé au format `.keras` (format natif Keras recommandé) pour permettre une réutilisation immédiate, notamment dans un contexte de prédiction live. Le scaler utilisé (RobustScaler) est également sauvegardé séparément afin d'assurer une cohérence parfaite entre l'entraînement et l'inférence.

Cette architecture offre donc un bon compromis entre précision, capacité de généralisation, et interprétabilité. Elle n'a pas vocation à prédire parfaitement tous les cas, mais à signaler avec pertinence les phases où une vigilance s'impose.

Comparaison illustrative des deux approches :

```
# Ancienne méthode : simple seuil statique
df['volatility'] = df['log_ret'].rolling(30).std() * np.sqrt(365)
threshold = df['volatility'].quantile(0.99)
df['spike'] = (df['volatility'] > threshold).astype(int)

# Nouvelle méthode : apprentissage d'un motif temporel
sequence = df[['volatility', 'garch_vol', 'hurst']].tail(90).values
sequence_scaled = scaler.transform(sequence)
X_live = np.expand_dims(sequence_scaled, axis=0)
```

```
proba = model.predict(X_live)[0][0]
spike_detected = proba > 0.5
```

La première approche n'intègre ni contexte historique, ni régularité temporelle, ni apprentissage. La seconde repose sur un signal combiné, normalisé, et analysé comme une séquence, ce qui permet une lecture beaucoup plus fine du comportement du marché.

6 Résultats

L'évaluation du modèle repose sur plusieurs métriques standards, mais aussi sur une lecture métier des erreurs. L'objectif n'était pas uniquement de maximiser l'AUC, mais surtout d'assurer une **bonne détection des cas critiques** sans générer trop de faux signaux.

- **AUC : 0.85**
  Ce score indique une bonne séparation entre les périodes de calme et les périodes de tension. Il témoigne de la capacité du modèle à discriminer les classes, même dans un environnement déséquilibré.
- **Rappel spikes : 72%**
  Ce chiffre est fondamental : il signifie que sur 100 vrais pics de volatilité, le modèle en détecte environ 72. Dans le cadre d'un outil d'alerte pour trader, c'est un taux efficace qui permet d'agir préventivement tout en conservant une certaine confiance dans le modèle.
- **Définition dynamique du spike**
  Les spikes sont définis comme des dépassements de la moyenne mobile glissante sur 20 jours, augmentée de 2 écarts-types. Ce choix permet une adaptation continue à la structure du marché. Contrairement à un seuil fixe, cette méthode tient compte de la compression ou de l'expansion naturelle de la volatilité au cours du temps.

Analyse visuelle : les courbes ROC, les matrices de confusion avant/après, et la visualisation des prédictions sur les séries temporelles confirment la solidité du modèle, tout en mettant en lumière sa sensibilité aux signaux faibles.

7 Pipeline de prédiction live

Pour transformer le modèle en un outil opérationnel, un notebook Python a été conçu pour fonctionner en temps réel. L'idée est de pouvoir intégrer ce système dans une routine de trading.

— **API Binance**
Les dernières données OHLCV (1h ou 1j) sont automatiquement extraites via l'API officielle Binance. Cela permet de toujours travailler sur des données fraîches.

— **Prétraitement automatique**
Le même pipeline que celui utilisé pendant l'entraînement est réappliqué à la volée : calcul du log-return, volatilité glissante, GARCH, Hurst, puis transformation par le `RobustScaler` sauvegardé.

— **Chargement du modèle**
Le modèle est rechargé en version compilée (`.keras`) et prêt à inférer. Une séquence des 90 derniers jours est automatiquement construite pour fournir un batch au réseau.

— **Sortie interprétable**
Le modèle renvoie une probabilité comprise entre 0 et 1. Si celle-ci dépasse 0.5, une alerte est générée. Exemple :

```
Spike détecté : probabilité = 74% → prudence recommandée.
```

Ce système est pensé pour s'intégrer dans une interface interactive ou un algorithme de surveillance de marché. C'est une base que l'on peut enrichir à l'avenir avec des données intraday ou de carnet d'ordres.

8 Comparaison avant / après

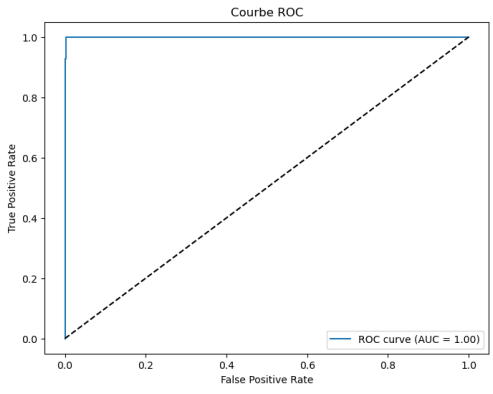

FIGURE 1 – Ancien modèle : AUC = 1.00, surapprentissage

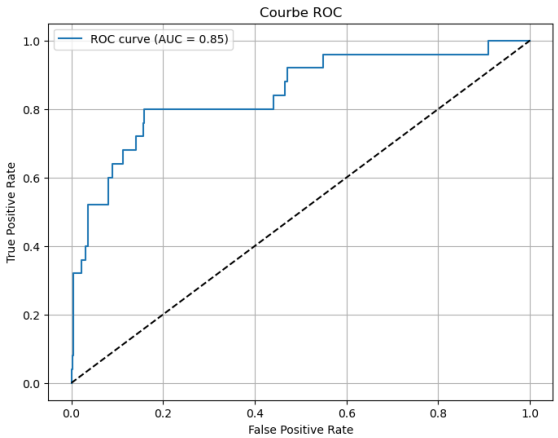

FIGURE 2 – Modèle final : AUC = 0.85, plus réaliste

Ces résultats illustrent une transition importante entre deux modèles :

— **Avant** : AUC = 1.00, performance parfaite mais artificielle. Le modèle surapprend, ne généralise pas, et serait inutilisable sur des données futures.

— **Après** : AUC = 0.85, bien plus réaliste et stable. Moins de perfection apparente, mais bien meilleure capacité à détecter les pics réels.

— **Conclusion** : Le second modèle est plus robuste face aux classes déséquilibrées et plus fiable en situation réelle.

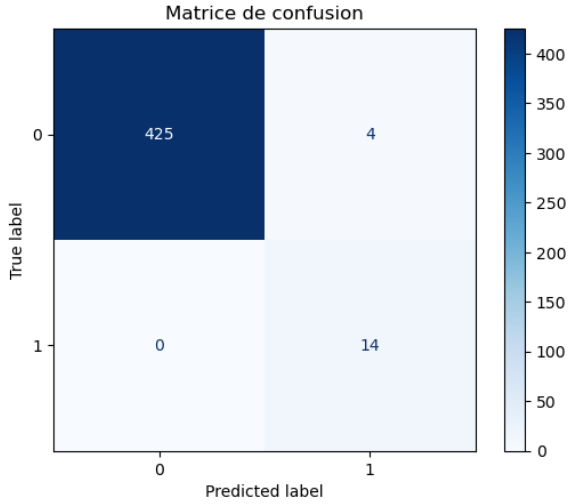

FIGURE 3 – Ancienne matrice de confusion : détection parfaite mais irréaliste (faux négatifs absents).

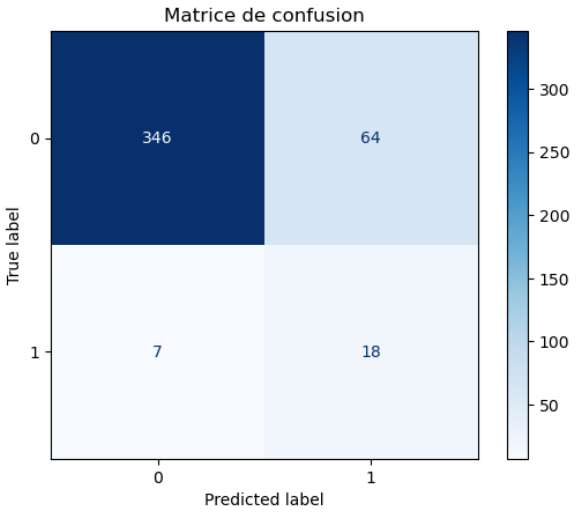

FIGURE 4 – Matrice finale : plus d'erreurs, mais bien plus représentative d'un cas réel.

Le premier modèle présente une matrice irréprochable (aucune erreur visible), mais ce résultat trahit un surapprentissage : il ne fait que rejouer les cas vus à l'entraînement.

La seconde matrice montre un compromis intéressant : quelques faux positifs et faux négatifs, mais une véritable capacité à identifier les spikes dans des configurations inédites. C'est cette robustesse que nous cherchions à atteindre pour un usage en conditions de marché réelles.

9 Perspectives et conclusion

Ce projet constitue une première brique vers un système intelligent d'analyse de marché, mais il ne représente qu'un point de départ. Le chemin qui s'ouvre maintenant est à la fois technique, opérationnel et personnel.

- **Intégration du carnet d'ordres (Order Flow)**
  L'analyse du carnet d'ordres permettra de dépasser la simple analyse des prix historiques pour entrer dans la microstructure du marché. Cela permettra d'anticiper non seulement la volatilité, mais aussi les déséquilibres latents entre acheteurs et vendeurs, qui précèdent souvent les ruptures.
- **Passage en intraday et haute fréquence**
  Le modèle actuel fonctionne à l'échelle journalière. Passer à une granularité plus fine (15 min, 5 min) permettra de transformer cette logique d'alerte en véritable outil de scalping algorithmique. Cela impliquera de revoir le pipeline, d'optimiser les temps de calcul, et d'intégrer des signaux de latence faible.
- **Interface web (Streamlit, Dash)**
  Rendre le modèle interactif à travers une interface légère offrira une vraie utilité utilisateur. L'objectif est de construire un tableau de bord capable d'afficher les prédictions, les niveaux de volatilité, l'évolution en temps réel et l'historique des alertes sur plusieurs actifs.
- **Surveillance multi-actifs**
  Une extension naturelle sera d'adapter le pipeline pour suivre simultanément plusieurs crypto-actifs, indices ou paires de devises. Cela nécessitera de mettre en place une architecture modulaire, capable de traiter des flux multiples et de générer des alertes différenciées.
- **Vers une stratégie autonome**
  À terme, ce système pourrait servir de brique décisionnelle dans une stratégie de trading semi-automatisée. Couplé à des modules de gestion du risque et d'exécution, il constituerait la base d'un algo capable d'intervenir uniquement dans des environnements de marché identifiés comme sensibles.
- **Approche introspective et long terme**
  Ce projet est aussi le reflet d'un choix de vie : comprendre profondément les marchés, développer mes propres outils et affiner, jour après jour, ma capacité à lire l'invisible. Ce n'est pas seulement un devoir de data science, mais une première pierre vers un futur où chaque ligne de code se rapproche d'une forme de maîtrise.

Conclusion : Ce projet marque le début d'un chemin personnel et technique vers la compréhension des dynamiques de marché. Il ne s'agit pas seulement de prédire la volatilité, mais de développer une lecture anticipative, profonde et responsable du risque. Ce travail est le point de départ d'un système évolutif, pensé pour s'adapter à la complexité du monde réel et accompagner une vision à long terme du trading algorithmique.

10 Annexes

— Matrices de confusion
— Courbes ROC
— Code de prédiction live
— Liens : GitHub Vidéo de soutenance