**Question Chatgpt** pour Introduction : choix de la méthodologie globale et de la direction de recherche

Je suis en train de travailler sur un projet de machine learning. Le sujet est le suivant : Construire un modèle d'apprentissage supervisé permettant de prédire si le prix de clôture du BTC sera plus haut ou plus bas qu'aujourd'hui. Le but étant de construire un modèle de trading simple afin d'acheter du btc si le modèle prédit une hausse et de vendre s'il prédit une baisse (chaque jour le BTC est vendu ou acheté à la même heure pour fermer le trade précédent). On ne souhaite pas ce soucier des problématique de take profit et de stop loss pour le moment. Propose des méthodologies de préparation de données, de stratégies du modèle, d'organisation de l'apprentissage permettant d'améliorer la performance du modèle.

**Réponse CHATGPT**

Créer un modèle de trading supervisé pour prédire si le prix de clôture du BTC sera plus haut ou plus bas le lendemain est une tâche intéressante et complexe. Voici une méthodologie détaillée pour préparer les données, choisir des modèles, et organiser l'apprentissage tout en cherchant à améliorer les performances du modèle.

**1. Préparation des données**

La qualité des données est cruciale pour de bons résultats en machine learning.

**a. Collecte des données**

* **Source** : Récupérez des données historiques du marché BTC/USDT ou BTC/USD depuis une API comme Binance, CoinGecko, ou Yahoo Finance.
* **Données à inclure** :
  + **Prix** : Ouverture, Haut, Bas, Clôture, Volume.
  + **Indicateurs techniques** : RSI, MACD, Moyennes mobiles (SMA/EMA), ATR, Bollinger Bands, etc.
  + **Facteurs exogènes** : Indicateurs macroéconomiques (inflation, taux d'intérêt), données liées à l'adoption des cryptos.

**b. Création de la cible**

* Transformez le prix de clôture en une variable binaire : df['target'] = (df['close'].shift(-1) > df['close']).astype(int)
  + 1 : Prix de clôture supérieur au prix actuel.
  + 0 : Prix de clôture inférieur ou égal.

**c. Ingénierie des caractéristiques**

* **Indicateurs techniques** :
  + RSI, MACD, Stochastique, Moyennes mobiles.
  + Fractales (patrons de chandeliers japonais).
* **Features temporels** :
  + Jour de la semaine, jour du mois.
* **Volumes** :
  + Moyennes mobiles sur les volumes.
* **Dérivées des prix** :
  + Rendements log (log returns), volatilité historique, momentum.

**d. Gestion des données temporelles**

* Divisez les données en séquences chronologiques pour éviter la fuite de données :
  + **Train** : Données anciennes (60-80%).
  + **Validation** : Données plus récentes (10-20%).
  + **Test** : Données les plus récentes (10-20%).

**e. Normalisation**

* Normalisez les caractéristiques numériques (par exemple, entre 0 et 1) pour stabiliser l'entraînement : from sklearn.preprocessing import MinMaxScaler
* scaler = MinMaxScaler()
* df\_scaled = scaler.fit\_transform(df)

**2. Choix et stratégies pour le modèle**

Le choix du modèle dépend de la complexité des données et de votre capacité à interpréter les résultats.

**a. Modèles simples pour un benchmark**

* **Régression logistique** : Simple et interprétable.
* **Random Forest** : Performant sur les données tabulaires avec peu de tuning.

**b. Modèles avancés**

* **Gradient Boosting Machines** : XGBoost, LightGBM, ou CatBoost.
* **Réseaux neuronaux récurrents (RNN)** : LSTM ou GRU pour capturer les dépendances temporelles.
* **Transformers** : Architectures comme les Time Series Transformers pour capturer les relations complexes.

**c. Métamodèles**

* Combinez plusieurs modèles avec un empilement (stacking) ou un vote majoritaire.

**d. Gestion du déséquilibre des classes**

* Si les hausses et baisses ne sont pas équilibrées, utilisez :
  + **Suréchantillonnage** avec SMOTE ou ADASYN.
  + **Pondération des classes**.

**3. Organisation de l'apprentissage**

Un apprentissage bien structuré améliore la performance et la robustesse.

**a. Validation croisée temporelle**

Utilisez une validation croisée spécifique aux séries temporelles, comme une **validation croisée en rolling window**. Exemple :

* Découpez les données en fenêtres temporelles successives.
* Entraînez sur la fenêtre passée et validez sur la prochaine fenêtre.

**b. Optimisation des hyperparamètres**

Utilisez des techniques comme **Grid Search**, **Random Search**, ou **Bayesian Optimization** pour ajuster les paramètres.

**c. Évaluation des performances**

* **Métriques** :
  + Précision, rappel, F1-score pour les classes 0 et 1.
  + Sharpe ratio : Mesure de la performance du modèle en termes de retour ajusté au risque.
* **Backtesting** :
  + Simulez le modèle sur des données historiques avec un backtest.
  + Vérifiez la performance (PNL, drawdown, taux de réussite).

**4. Améliorations potentielles**

**a. Augmentation des données**

* Ajouter des données exogènes comme les tweets d'Elon Musk ou les nouvelles sur les régulations.
* Augmenter les features en analysant d'autres cryptos liées (ETH, BNB).

**b. Gestion des tendances**

* Intégrer des caractéristiques de détection de tendance (trend detection) pour éviter les biais dans les marchés plats.

**c. Analyse des erreurs**

* Analysez les faux positifs et faux négatifs pour identifier des patterns spécifiques.

**d. Régularisation et dropout**

* Pour éviter le surapprentissage :
  + Utilisez L1/L2 régularisation ou dropout pour les réseaux neuronaux.

**5. Exemple d'implémentation avec XGBoost**

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

# Séparer les données

X = df.drop(columns=['target'])

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

# Créer et entraîner le modèle

model = xgb.XGBClassifier()

model.fit(X\_train, y\_train)

# Évaluer le modèle

y\_pred = model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

Cette méthodologie vous permet de structurer votre projet tout en explorant des pistes pour améliorer les performances. N'hésitez pas à affiner selon les résultats obtenus.

**Question CHATGPT:**

Tu as proposé plusieurs modèles simples et complexes. Lesquels seraient les plus performant lorsqu'il s'agit de prédire une variable binaire sur des données temporelles. En effet, la prédiction de la variable binaire ne dépend certainement pas seulement des valeurs des variables pour la journée n mais plutôt de l'ensemble du jeu de données. Quel modèle permet de prendre en compte les trend, et repérer les potentiels patterns antérieurs indiquant une hausse ou une baisse

**Réponse CHATGPT:**

Pour prédire une variable binaire sur des données temporelles comme les prix du BTC, il est crucial d'utiliser des modèles capables de capturer les tendances, patterns, et dépendances temporelles. Voici une analyse des modèles les plus pertinents pour votre cas d'usage, en tenant compte de leur capacité à exploiter les données historiques et à détecter des tendances.

**1. Modèles les plus adaptés pour des données temporelles**

**a. Modèles avancés de machine learning traditionnels**

Ces modèles nécessitent une préparation soignée pour inclure les tendances et patterns historiques.

1. **Gradient Boosting Machines (XGBoost, LightGBM, CatBoost)**
   * **Avantages** :
     + Performants sur des données tabulaires.
     + Capturent les interactions non linéaires entre les variables.
     + Permettent d'intégrer manuellement des features basées sur les tendances (par exemple, moyennes mobiles, momentum, volatilité).
   * **Limite** :
     + Ne capturent pas nativement les dépendances temporelles sans ingénierie des caractéristiques.
2. **Astuce** : Intégrer des caractéristiques temporelles (lagged features, moyennes mobiles, etc.) permet à ces modèles d'approcher les performances des modèles séquentiels.
3. **Random Forest**
   * Moins performant que les modèles boostés, mais utile pour un benchmark.
   * Idéal si l'on dispose d'un ensemble limité de caractéristiques bien choisies.

**b. Modèles séquentiels basés sur les réseaux neuronaux**

Les modèles ci-dessous capturent les tendances et patterns sans nécessiter autant de travail d'ingénierie des caractéristiques.

1. **Long Short-Term Memory (LSTM)**
   * **Avantages** :
     + Conçus spécifiquement pour les séries temporelles.
     + Capturent les dépendances à long terme et court terme.
   * **Cas d'usage** :
     + Détection de motifs récurrents.
     + Analyse de séquences longues (par exemple, 30 jours de données pour prédire un mouvement).
2. **Limite** :
   * Plus lent à entraîner.
   * Nécessite un plus grand volume de données.
3. **Gated Recurrent Unit (GRU)**
   * Similaire à LSTM, mais avec moins de paramètres, ce qui le rend plus rapide à entraîner.
   * Adapté si vous avez moins de données ou si vous privilégiez des modèles plus légers.
4. **Transformers pour séries temporelles**
   * **Avantages** :
     + Excellents pour capturer des patterns complexes dans des séries longues.
     + Modèles modernes, performants dans de nombreux benchmarks récents.
   * **Exemple** :
     + Modèles comme TimeSeriesTransformer ou Informer.
   * **Limites** :
     + Complexité de mise en œuvre.
     + Exigent une puissance de calcul élevée.

**2. Comparaison pour capturer les tendances**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modèle** | **Dépendances temporelles** | **Complexité d'implémentation** | **Données requises** | **Performance attendue** |
| XGBoost/LightGBM | Moyenne (avec features lag) | Moyenne | Faible à modérée | Bonne sur courtes séquences |
| LSTM | Très bonne | Élevée | Modérée à élevée | Très bonne |
| GRU | Bonne | Moyenne | Modérée | Bonne |
| Transformers | Excellente | Très élevée | Élevée | Excellente |

**3. Prise en compte des tendances et patterns**

**Modèles basés sur des réseaux neuronaux (LSTM, GRU, Transformers)**

* Ces modèles apprennent directement les dépendances temporelles et les tendances, sans nécessiter de créer manuellement des indicateurs comme les moyennes mobiles ou les rendements différés.

**Exemple avec LSTM**

import numpy as np

import pandas as pd

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

from sklearn.preprocessing import MinMaxScaler

# Charger les données

df = pd.read\_csv('btc\_prices.csv') # Par exemple : colonnes [open, high, low, close, volume]

# Préparation des données

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(df[['close']].values)

# Créer les séquences pour LSTM

sequence\_length = 30 # Utiliser les 30 derniers jours pour prédire

X, y = [], []

for i in range(sequence\_length, len(scaled\_data)):

X.append(scaled\_data[i-sequence\_length:i, 0])

y.append(1 if scaled\_data[i, 0] > scaled\_data[i-1, 0] else 0)

X, y = np.array(X), np.array(y)

X = np.reshape(X, (X.shape[0], X.shape[1], 1)) # Reshape pour LSTM

# Construire le modèle LSTM

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units=50))

model.add(Dropout(0.2))

model.add(Dense(units=1, activation='sigmoid')) # Prédiction binaire

# Compiler et entraîner le modèle

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(X, y, epochs=10, batch\_size=32)

**4. Recommandations**

1. **Pour des données limitées ou un démarrage rapide** : Utilisez **XGBoost** ou **LightGBM** avec des indicateurs techniques soigneusement choisis.
2. **Pour capturer des dépendances temporelles complexes** :
   * LSTM ou GRU si vous avez une infrastructure modérée.
   * Transformers si vous avez une infrastructure puissante et un grand volume de données.
3. **Approche hybride** :
   * Combinez XGBoost pour intégrer des features explicites avec LSTM pour capturer des dépendances profondes.

En résumé, pour capturer les tendances et patterns historiques, les **modèles séquentiels (LSTM, GRU)** et **Transformers** sont les meilleurs choix. Si vous privilégiez la simplicité et la rapidité, les modèles boostés comme XGBoost sont de bons candidats avec des indicateurs manuellement intégrés.

**Question CHATGPT:**

tu as beaucoup parlé du coût de calcul des modeles LTSM ou Transformers. Je suis sur un mac M1 et mes données sont celles du prix du BTC depuis 28 juillet 2023 pour le moment. Est ce que mon ordi est capable d'obtenir des résultats convaincant avec ces modèles ou la puissance requise est trop élevée. De plus, que proposerais tu à propos du choix des données: je souhaite prédire et prendre la décision d'achat ou de vente tous les jours à minuit, serait il interessant de prendre les données 1h plutot qu'en 1D, dois-je garder mon grand jeu de données de 500 jours ou plutot découpé celui en période temporelle?

**Réponse CHATGPT:**

Travailler avec des modèles comme LSTM ou Transformers sur un Mac M1 avec un dataset limité comme le vôtre (environ 500 jours de données) est tout à fait envisageable. Voici une réponse détaillée à vos deux préoccupations : **la faisabilité technique** et **le choix des données**.

**1. Faisabilité technique sur un Mac M1**

Le Mac M1 est équipé de cœurs CPU/GPU performants et prend en charge les bibliothèques optimisées pour le machine learning, comme TensorFlow et PyTorch. Avec un dataset modeste comme le vôtre, voici quelques points à considérer :

**Modèles LSTM ou GRU**

* **Faisabilité** : Les LSTM ou GRU sont tout à fait réalisables sur un Mac M1 avec un jeu de données de taille modérée comme le vôtre. Ces modèles sont moins gourmands en mémoire et en puissance de calcul que les Transformers.
* **Temps d’entraînement attendu** : En fonction de la longueur des séquences (par exemple, 30 ou 60 jours) et du nombre d’unités dans les couches, l’entraînement devrait prendre quelques minutes à quelques heures.
* **Astuce** : Réduisez la taille des couches LSTM (50 à 100 unités par couche suffisent généralement) et utilisez des batchs de taille modérée (par exemple, 32 ou 64).

**Modèles Transformers**

* **Faisabilité** : Transformers sont plus gourmands en calcul, mais avec un dataset de 500 jours et des séquences courtes (30 à 60 jours), vous pouvez les entraîner en limitant leur taille.
* **Stratégies pour les rendre légers** :
  + Utilisez un modèle comme Informer ou TimeSeriesTransformer (moins complexes que GPT-like Transformers).
  + Limitez la profondeur (2 ou 3 couches) et les dimensions des embeddings.

En résumé, vous pouvez obtenir des résultats convaincants avec des modèles séquentiels, mais préférez les **LSTM/GRU** pour leur simplicité et leur faible coût.

**2. Choix des données pour améliorer la prédiction**

**a. Fréquence des données : 1H vs 1D**

La décision d’utiliser des données horaires (1H) ou journalières (1D) dépend de la volatilité et de la granularité souhaitée dans votre modèle de trading.

**Données journalières (1D)**

* **Avantages** :
  + Moins de bruit : Les données journalières filtrent les fluctuations intrajournalières volatiles.
  + Simplicité : Plus faciles à traiter, moins gourmandes en mémoire.
  + Convient si vous prenez des décisions **quotidiennes** à minuit.
* **Inconvénients** :
  + Perte d’informations sur les mouvements intrajournaliers (spikes, patterns spécifiques).

**Données horaires (1H)**

* **Avantages** :
  + Capturent des patterns intrajournaliers, utiles pour des décisions basées sur la volatilité à court terme.
  + Plus de données pour entraîner le modèle (24x plus pour 500 jours = environ 12,000 points).
* **Inconvénients** :
  + Plus de bruit dans les données.
  + Plus gourmand en calcul et en mémoire.
  + Risque de surapprentissage si les patterns intrajournaliers ne sont pas significatifs pour la prédiction quotidienne.

**Recommandation :**

* Si votre objectif est de prédire à **minuit chaque jour**, restez sur des données journalières (1D).
* Si vous voulez tester des modèles plus fins ou intégrer des patterns intrajournaliers, combinez les deux fréquences en créant des **features agrégées** (par exemple, la moyenne, le max, et le min des prix horaires sur 24 heures).

**b. Taille du dataset : 500 jours vs périodes découpées**

Le choix de conserver tout le dataset ou de le découper dépend de la cohérence des patterns dans vos données.

**Conserver 500 jours complets**

* **Avantages** :
  + Modèle mieux entraîné avec une vue plus globale des tendances historiques.
  + Les patterns de tendance et de saisonnalité (comme les mouvements mensuels) sont capturés.
* **Inconvénients** :
  + Si les marchés ont changé récemment (par exemple, changement de volatilité), les données anciennes peuvent fausser le modèle.

**Découper en périodes temporelles récentes**

* **Avantages** :
  + Le modèle se concentre sur les données les plus récentes et pertinentes.
  + Réduit le risque que les données plus anciennes diluent les patterns actuels.
* **Inconvénients** :
  + Moins de données, ce qui peut entraîner un surapprentissage.
  + Perte des tendances à plus long terme.

**Recommandation :**

* **Stratégie principale** : Entraînez sur les **500 jours complets** et testez sur des données récentes (par exemple, les 50 derniers jours).
* **Stratégie alternative** : Si vous observez des changements récents dans le comportement du marché, testez un découpage avec les **200-300 derniers jours** pour entraîner le modèle.

**3. Suggestions supplémentaires**

1. **Créer des features explicatives (pour XGBoost ou LSTM) :**
   * Indicateurs techniques (RSI, MACD, moyennes mobiles, volatilité).
   * Features temporelles : jour de la semaine, effet de fin de mois.
   * Agrégations intrajournalières si vous utilisez des données horaires.
2. **Gestion des tendances macroéconomiques :**
   * Ajoutez des données externes comme l’indice S&P 500, les taux d’intérêt ou les volumes de transactions, si disponibles.
3. **Approche incrémentale pour les périodes temporelles :**
   * Entraînez votre modèle de façon **rolling** : utilisez une fenêtre glissante sur les 300 derniers jours pour l’entraînement et prédisez les jours suivants.
   * Cela permet au modèle de s’adapter aux récents changements de marché.

**Conclusion**

Avec un Mac M1 et un dataset de 500 jours, les modèles LSTM ou GRU sont tout à fait réalisables pour des prédictions quotidiennes. Privilégiez les données journalières (1D) pour simplifier le traitement et limiter le bruit, tout en gardant les 500 jours pour capturer les tendances à long terme. Si vous souhaitez explorer davantage, intégrer des données horaires (1H) sous forme de features agrégées peut enrichir le modèle sans compromettre la performance.

L’approche dépend de vos tests initiaux : commencez simple avec des données journalières, puis affinez en introduisant davantage de granularité et des modèles plus complexes si nécessaire.