

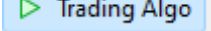
Bonjour,

J'espère que vous allez bien,

Ceci est un code pour trader BTC sur unité de temps de 15 minutes.

Le code est divisé en 3 parties,

- **main.py**, le fichier principal de ce projet, responsable de l'exécution globale du code. Il est utilisé pour coordonner l'exécution des différentes parties du programme, telles que l'initialisation, le chargement des données, le traitement des données, l'entraînement du modèle, l'évaluation du modèle, etc. En bref, c'est là que l'ensemble du processus est orchestré.
- **Settings.json**, c'est là où il y a les informations d'identification pour Metatrader5, j'ai gardé mes informations (compte Demo) au cas où vous n'utilisez pas Metatrader5.
- Sur **Mt5 lib**, il y a les fonctions de calculs des indicateurs, incluent l'initialisation de MetaTrader5, la récupération des bougies, le calcul des indicateurs techniques tels que le RSI, SuperTrend, Resistance support, higher highs, higher lows, etc., ainsi que des fonctions pour le traitement des données et la préparation des données d'entraînement pour les modèles de machine learning.

Pour faire fonctionner le bot, il suffit de télécharger MetaTrader5, activé  sur le logiciel Metatrader5 et lancé l'algorithme.

Comme j'ai mentionné précédemment, j'ai gardé mes identifiants au cas où vous n'utilisez pas MetaTrader5 et vous ne disposez pas d'un compte Demo.

```
{ "mt5":  
  { "username": "510984",  
    "password": "Abcd123456@",  
    "server": "Cryptograph-Live",  
    "mt5_pathway": "C:\\Program Files\\MetaTrader 5\\terminal64.exe",  
    "symbols": "BTCUSD",  
    "timeframe": "M15"}  
}
```

I- Récupération des données bougies

```
if __name__ == '__main__':
    print("Let's build a bot")

    # Load project settings
    project_settings = get_project_settings(import_filepath=setting_filepath)

    # Start MT5 and initialize symbols
    startup = start_up(project_settings=project_settings)

    # Extract timeframe and symbols from settings
    timeframe = project_settings['mt5']['timeframe']
    symbol = project_settings['mt5']['symbols']

    # Get candlestick data
    df = mt5_lib.get_candlesticks(
        symbol=symbol,
        timeframe=timeframe,
        number_of_candles=1000)
```

L'algorithme utilise la fonction Get_candlesticks pour récupérer les données de bougies

II- Calcul des indicateurs

L'algorithme récupère les données de Metatrader5, sur un laps de temps de 15 minutes et effectue un calcul des différents indicateurs :

```
discrete_features = [
    "RSI_Over_70",
    "RSI_Under_30", "VWAP_Cross_From_Above", "SUPER_UP",
    'HIGH', 'LOW', 'ADX_UP', 'ADX_DOWN', 'DOJI', 'Chaiken_Signal', "14EMA/21EMA_Cross", "14EMA>21EMA", "MACD"
]

st, ut, dt = mt5_lib.get_supertrend2(df['high'], df['low'], df['close'], lookback=14, multiplier=3)
df = df[~df.index.duplicated()]
ut = ut[~ut.index.duplicated()]
dt = dt[~dt.index.duplicated()]
st = st[~st.index.duplicated()]
df = pd.concat([df, ut, dt, st], axis=1)
pivots = mt5_lib.get_resistance_support2(df)
df = mt5_lib.distance_to_close_supres(pivots, df)
df = mt5_lib.ADX_data(df, adx_period=14)

df['Chaikin'] = mt5_lib.chaikin_oscillator(df, fast_period=3, slow_period=10)
df = mt5_lib.CCI(df)

df = mt5_lib.EMA(df)
df = mt5_lib.ATRMACD(df)
df = mt5_lib.Signal(df)
df = mt5_lib.Linreg(df)
df = mt5_lib.German(df, historical_period=5000)
```

Comme indiqué sur le screen, les différents indicateurs utilisés pour la Machine Learning sont RSI, VWAP, High, Low, SuperTrend, ADX, Chaiken, et EMA, MACD, Lower Lows HigherHighs, Candle supérieur à un niveau (500).

Je me suis mis sûr qu'il n'y a pas une fuite de données notamment sur les Lower Lows et Higher Highs, pour se faire j'ai retardé les résultats de 5 bougies pour éviter toute sorte de fuite de donnée.

Si l'indicateur donne un indice de d'achat, il prend 1, si un indice de vente, il prend 0. Cela principalement pour éviter le problème d'Overfitting.

III- Préparation pour Machine Learning :

Pour optimiser les entrées, l'algorithme garde seulement les valeurs où il y avait un break-out de Bollinger Bands

```
filtered_df = df[((df['Bollinger_Bands_Above_Upper_BB'] != 0) |  
(df['Bollinger_Bands_Below_Lower_BB'] != 0))]
```

Ensuite, une fois que les données ont été filtrées, l'algorithme prépare les données en divisant le DataFrame filtré en ensembles d'entraînement et de test. Il utilise un ratio de 80% pour l'ensemble d'entraînement et de 20% pour l'ensemble de test. Cela peut être réalisé à l'aide de différentes méthodes, telles que `train_test_split` de scikit-learn ou simplement en sélectionnant les premières lignes pour l'ensemble d'entraînement et les dernières lignes pour l'ensemble de test.

IV- Machine Learning :

La fonction Machine Learning, fait une optimisation du variable Random State – cette optimisation peut être sauté, en Hard coding le Random state à 1.

La fonction `TPSL` calcule les niveaux de stop-loss et de take-profit pour chaque position de trading en se basant sur une stratégie de gestion des risques prédéfinie (3 Take profit, 2 stop loss, un RR de 1.5).

Elle initialise des colonnes dans le DataFrame pour stocker ces valeurs ainsi que d'autres informations relatives à la sortie des positions. En utilisant le prix de clôture actuel et l'ATR associé, elle détermine les niveaux de stop-loss et de take-profit adaptés pour les positions longues et courtes.

Ensuite, elle parcourt les données suivantes pour chaque position afin de vérifier si les conditions de sortie sont remplies, mettant à jour les colonnes de sortie en conséquence.

Une fois toutes les positions traitées, elle retourne le DataFrame mis à jour, excluant les premières lignes qui pourraient ne pas avoir suffisamment de données pour le calcul.

- ⇒ Si machine learning prédit correctement l'entrée, elle utilise le Take_profit calcul, si elle prédit incorrectement l'entrée, elle prend le stop_loss.
- ⇒ A Noter bien, que ma fonction elle marche seulement si le Takeprofit > Stoploss, si c'est le contraire, le calcul sera incorrect

Finalement pour Machine Learning, j'ai utilisé Decision Tree avec les paramètres Classique.

L'algorithme donne des bons résultats sur le long terme, mais je suis conscient que beaucoup d'amélioration peuvent être apportées notamment l'utilisation de support et résistances et High/Low/Open/Close des journées précédentes. Je préfère si je veux trader le BTC, j'essaie de l'utiliser en Semi-Manuel, c'est-à-dire j'active et je désactive Long-Short, en se basant sur mes analyses du marché en temps réel.

De plus, le code peut être beaucoup mieux organisé dans la mesure où on pourrait au lieu d'une méthodologie fonctionnelle, adopter une méthodologie de gestion d'objet et d'attribut qui pourrait nous faciliter ensuite le maintien du code

PS : L'algorithme prend du temps pour analyser 90 000 candle. Sur le live account, j'utilise :

```
joblib.dump(discrete_rus, 'random_forest_model.pkl')
```

Pour enregistrer le résultat de la machine learning et optimiser rapidement les entrées.

Pour n'importe quelle demande/clarification ou conseils n'hésitez pas à me contacter sur taha.ettouhami@rennes-sb.com.

Agréable lecture,

Taha