ДОДАТОК А

Вихідний код програми

IntelliBotStrategy.py

```
import numpy as np
import pandas as pd
from pandas import DataFrame
import talib.abstract as ta
import freqtrade.vendor.qtpylib.indicators as qtpylib
import joblib
import os
from ai_module import get_predictor
from ai_module_v4 import AIModuleV4
use_exit_signal = True
    exit_profit_only = False
    ignore_roi_if_entry_signal = False
buy_rsi = IntParameter(low=1, high=50, default=30,
space='buy', optimize=True, load=True)
    sell_rsi = IntParameter(low=50, high=100, default=70,
space='sell', optimize=True, load=True)
    short_rsi = IntParameter(low=51, high=100, default=70,
space='sell', optimize=True, load=True)
    exit_short_rsi = IntParameter(low=1, high=50,
default=30, space='buy', optimize=True, load=True)
startup_candle_count: int = 200
order_types = {
        'entry': 'limit',
        'exit': 'limit',
        'stoploss': 'market',
        'stoploss_on_exchange': False
    }
order_time_in_force = {
        'entry': 'GTC',
        'exit': 'GTC'
```

```
        }
        plot_config = {
            'main_plot': {
                'tema': {},
                'sar': {'color': 'white'},
            },
            'subplots': {
                "MACD": {
                    'macd': {'color': 'blue'},
                    'macdsignal': {'color': 'orange'},
                },
                "RSI": {
                    'rsi': {'color': 'red'},
                }
            }
        }
        def __init__(self, *args, **kwargs):
            super().__init__(*args, **kwargs)
            from ai_module_v4 import AIModuleV4
            self.ai_module = AIModuleV4()
            self.predictor = get_predictor("ai")
            self.model                                          =
joblib.load("user_data/strategies/intellibot_model_v3.pkl")
            self.scaler                                         =
joblib.load("user_data/strategies/intellibot_scaler.pkl")
      def   populate_indicators(self,   dataframe:   DataFrame,
metadata: dict) -> DataFrame:
            # Ensure all expected keys exist for TA-Lib
            dataframe['close'] = dataframe['close'] if 'close'
in dataframe else dataframe['Close']

            # Індикатори для моделі
            dataframe['EMA_20']      =      ta.EMA(dataframe,
timeperiod=20)
            dataframe['EMA_50']      =      ta.EMA(dataframe,
timeperiod=50)
```

```
            dataframe['ROC']        =        ta.ROC(dataframe,
timeperiod=10)
            dataframe['bb_bbm'] = dataframe['bb_middleband']
            dataframe['bb_bbh'] = dataframe['bb_upperband']
            dataframe['bb_bbl'] = dataframe['bb_lowerband']
            # Bollinger Width
            bollinger                                          =
qtpylib.bollinger_bands(qtpylib.typical_price(dataframe),
window=20, stds=2)
            dataframe['bb_lowerband'] = bollinger['lower']
            dataframe['bb_middleband'] = bollinger['mid']
            dataframe['bb_upperband'] = bollinger['upper']
            dataframe["bb_width"] = (
                    (dataframe["bb_upperband"]                -
dataframe["bb_lowerband"]) / dataframe["bb_middleband"]
            )
            dataframe['RSI'] = ta.RSI(dataframe)
       def  populate_entry_trend(self,  dataframe:  DataFrame,
metadata: dict) -> DataFrame:
            if dataframe.empty:
                return dataframe
            dataframe['ai_signal']                             =
self.ai_module.predict(dataframe)
            dataframe.loc[dataframe['ai_signal']     ==     0,
'enter_long'] = 1
            dataframe.loc[dataframe['ai_signal']     ==     2,
'exit_long'] = 1
            dataframe['SMA_10']      =      ta.SMA(dataframe,
timeperiod=10)
            dataframe['EMA_10']      =      ta.EMA(dataframe,
timeperiod=10)
            indicator_columns = ['SMA_10', 'EMA_10', 'EMA_20',
'EMA_50', 'RSI', 'ROC', 'macd',
                                 'bb_middleband',
'bb_upperband', 'bb_lowerband', 'bb_width']
```

```
            mask                                                =
dataframe[indicator_columns].notna().all(axis=1)
            features = dataframe.loc[mask, indicator_columns]
            prediction = self.ai_module.predict(features)
            dataframe.loc[mask, 'ai_signal'] = prediction
            dataframe.loc[dataframe['ai_signal']     ==     0,
'enter_long'] = 1  # BUY
            dataframe.loc[dataframe['ai_signal']     ==     2,
'exit_long'] = 1  # SELL
            return dataframe
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	100 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Приходченко К.О.			Використання патернів програмування для розробки інтелектуального трейдінг бота	Шифр групи	Код напр./спец.
Перевірив		Кудрявцева М.С.				ІТШІ-21-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					