Trader bot

ربات معامله‌گری مبتنی بر هوش مصنوعی، با بهره‌گیری از الگوریتم‌های پیشرفته و قدرت تحلیل داده‌های بازارهای مالی، توانایی برنامه‌ریزی و اجرای معاملات را به صورت خودکار دارد. این ربات با دقت و سرعت بالا، به تجزیه و تحلیل روندها و الگوهای بازار مالی می‌پردازد و بر اساس آن‌ها تصمیم‌گیری در خصوص معاملات را انجام می‌دهد. با استفاده از انواع این ربات شامل quant و advance quant، سرمایه‌گذاران می‌توانند با اطمینان بیشتری در بازارهای مالی فعالیت نمایند و از مزایای قابل توجهی همچون دقت بالا، سرعت عمل و بازدهی بیشتر بهره‌مند شوند.

```
def bot(bot_model, initialize, currency='EURUSD', TimeFrame='15m',
        risk=100, max_position_time='3*TimeFrame', max_pending_time='0.1*max_time',
        use_haiken_ashi=False, hyper_tune=False, plot_results=False, force_to_train=False):
    """
    Run a trading bot based on the specified model.

    Parameters:
    - bot_model (str): The type of trading bot model to use. Should be 'quant' or 'advance_quant'.
    - initialize (function): The initialization function for the trading bot.
    - currency (str): The currency pair to trade. Default is 'EURUSD'.
    - TimeFrame (str): The time frame for the trading data. Default is '15m'.
    - risk (float): The risk amount for each trade. Default is 100.
    - max_position_time (str): The maximum time allowed for a position to be open. Default is '3*TimeFrame'.
    - max_pending_time (str): The maximum time allowed for a pending order to be active. Default is '0.1*max_time'.
    - use_haiken_ashi (bool): Whether to use Haiken Ashi candles for the trading bot. Default is False.
    - hyper_tune (bool): Whether to perform hyperparameter tuning for the trading bot. Default is False.
    - plot_results (bool): Whether to plot the trading results. Default is False.
    - force_to_train (bool): Whether to force the trading bot to train even if pre-training data is available. Default is False.

    Returns:
    - info (dict): Information about the trading bot run.
    - quant_outputs (dict): Outputs from the trading bot.

    Raises:
    - ValueError: If an invalid bot model is specified.
    """
```

Quant Model

پیش از هرچیز ابتدا نیاز است که دادگان گذشته ارز مدنظر از پلتفرم متا تریدر برای تجزیه تحلیل های اتی استخراج شود. برای این منظور در قسمت کد get_data.py تابع get_data_from_mt5 مورد استفاده قرار می‌گیرد.

```
## Download historical market data from MetaTrader5

def get_data_from_mt5(initialize, Ticker, TimeFrame):
    """
    Download historical market data from MetaTrader5.

    Parameters:
        initialize: A list containing the login credentials and server information for the MetaTrader5 account.
        The list should be in the format [login, password, server].
        Ticker: A string representing the currency ticker to download.
        TimeFrame: A string representing the time frame of the data to download.
        Valid values are "1m", "5m", "15m", "30m", "1h", "4h", "1d", "1w".

    Returns:
        A pandas DataFrame containing the historical market data.

    Examples:
        # Download historical market data from MetaTrader5
        initialize = [123456, 'password', 'MetaTraderServer']
        Ticker = 'EURUSD'
        TimeFrame = '1h'
        data = get_data_from_mt5(initialize, Ticker, TimeFrame)
    """
```

لازم به ذکر است که در کد فوق باید interval قیمت نهایی اصلاح شود. زیرا اگر در تایم فریم ۱۵ دقیقه، در ساعت ۱۱:۵۴ دیتا را دریافت کنیم، قیمت close آخرین داده برابر با مقدار قیمت در آن لحظه و قیمت close قبلی برابر با قیمت در ساعت ۱۱:۴۵ دقیقه است و interval زمانی آخرین دیتا برابر با ۹ دقیقه است که با ۱۵ دقیقه برابر نیست. بدین منظور تابع correct candle برای اصلاح این خطا استفاده می‌شود.

```
def correct_candle(initialize, ticker, timeframe):
    """
    Resamples the OHLC data of a currency pair based on the desired time interval and start time.

    Parameters
    ----------
    initialize : list of str
        List containing the login, password, and server information for the trading account.
    ticker : str
        String representing the currency pair to be analyzed (e.g. 'EURUSD', 'GBPJPY', 'XAUUSD').
    timeframe : str
        String representing the desired time interval for the resampled data, chosen from the keys of the interval dictionary
        (e.g. '5m', '15m', '30m', '1h', '4h').

    Returns
    -------
    pandas DataFrame
        A DataFrame containing the resampled OHLC data, along with the mean price, trading volume, and additional information
        from the original data.

    """
```

ایده اصلی در مدل quant این است که فرض می‌شود داده‌های گذشته جفت ارز که تشکیل یک سری زمانی y_t می‌دهند متشکل از ترم های زیر است:

$$y_t = \tau_t + s_t + r_t$$

که در آن τ_t روند خطی، s_t ترم فصلی، r_t ترم غیر خطی می‌باشد.

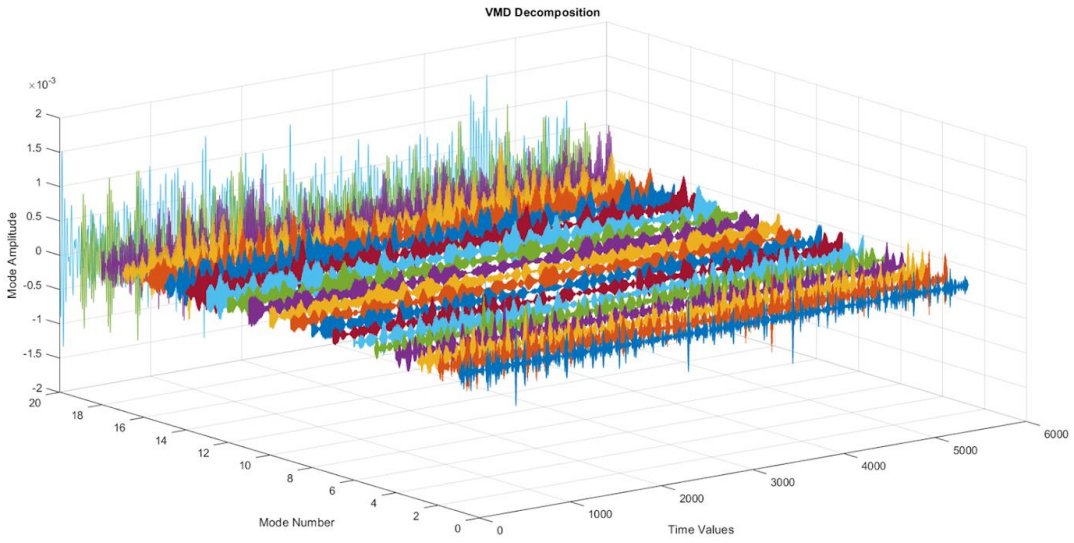
برای پیش بینی y_{t+1} مراحل زیر طی می‌شود:

- با استفاده از رگرسیون خطی، روند خطی محاسبه می‌شود
- با استفاده از تبدیل فوریه، ترم فصلی تخمین زده می‌شود

خروجی این مرحله با استفاده از تابع Fourier در کد utils.py:

```
def fourier(data, forecast_horizen):

    """
    Perform long-term forecasting using Fourier transform.

    Parameters:
        data: A 1D numpy array containing the input data.
        forecast_horizon: An integer representing the number of time steps to forecast.

    Returns:
        A 1D numpy array containing the forecasted values.

    The function first estimates a linear trend in the input data using ordinary least squares (OLS). The trend is then subtracted from the data to obtain a detre

    The forecasted values are then added back to the linear trend to obtain the final forecast.

    Examples:
        # Perform long-term forecasting using Fourier transform
        data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
        forecast_horizon = 5
        forecast = Forecast(data, forecast_horizon)
    """
```

برای پیش بینی ترم غیر خطی $r_t = y_t - (s_t + r_t)$، ابتدا برای ایستا سازی مشتق گرفته می‌شود. سپس ترم ایستا سازی شده به چندین مولفه فرکانسی با استفاده از الگوریتم Variational Mode Decomposition تجزیه می‌شود. مدهای تجزیه شده به عبارت دیگر فضای ویژگی مورد استفاده مدل یادگیری ماشین هستند که با استفاده از تاخیرهای آن مدل می‌تواند ترم غیر خطی را نیز تخمین بزند.



مراحل فوق در تابع quant_bot به صورت زیر اجرا می‌شود:

```
data = df['Mean'].to_numpy()
n = len(df)

fft_forecast = fourier(data, int(0.1 * n))

y = data - fft_forecast[:n]
y_diff = np.diff(y)

Target = y_diff[-n_sample:]

imfs = VMDdecomposition(Target, Numimf, plot_results)

X_scaled,_ = standardization(imfs.to_numpy())

target, scaler = standardization(Target.reshape(-1,1))

Inputs, X_forecast = FindBestLags(X_scaled,
                                  target.reshape(-1,),
                                  lagmax=lagmax,
                                  nlags=nlags)
```

که در آن تابع FindBestLags و standardizations به صورت زیر تعریف می شوند:

```python
def standardization(Data):
    """
    Standardize data to have zero mean and unit variance.
    """

    transformer = StandardScaler()
    standardized_data = transformer.fit_transform(Data)
    return standardized_data, transformer
```

```python
def FindBestLags(Inputs, Targets, lagmax=200, nlags=10):
    """
    Selects the best lags for each input feature based on their correlation with the target variable.

    Parameters:
    -----------
    Inputs : array-like, shape (n_samples, n_features)
        The input features.
    Targets : array-like, shape (n_samples, )
        The target variable
    lagmax : int, optional (default=200)
        The maximum lag to consider for the input features.
    nlags : int, optional (default=10)
        The number of lags to select for each input feature.

    Returns:
    --------
    Best_Inputs : array-like, shape (n_samples, nlags, n_features)
        The lagged input features selected based on the best lags.

    Forecast_Inputs : array-like, shape (2, nlags, n_features)
        The lagged input features for the last two time steps, used for forecasting.

    """
```

روش LightGBM به دلیل سرعت بالا و دقت پیش‌بینی بالاتر به عنوان مدل مورد استفاده قرار گرفته است. و فرپارامترهای آن با بهینه سازی طبق خطا تست در تقسیم بندی های مختلف بدست می‌آیند. نمونه نتایج این روش در زیر نشان داده شده است:

لازم به ذکر است روش فوق میانگین قیمت را پیش‌بینی می‌کند و برای پیدا کردن حد ضرر و سود از استراتژی زیر استفاده می‌شود:

اگر بر طبق پیش‌بنی بنا باشد معامله خرید انجام بدهیم، حد سود را کمی پایین تر از بیشینه قیمت قرار می‌دهیم و حد ضرر را کمی پایین تر از قیمت کمینه، حال قیمت پیشینه به صورت زیر تخمین زده می‌شوند:

$$\hat{y}_{max_{t+1}} = \hat{y}_{avg_{t+1}} + \alpha * moving\ average\left(\hat{y}_{max_t} - \hat{y}_{avg_t}\right) + \beta * moving\ std\ \left(\hat{y}_{max_t} - \hat{y}_{avg_t}\right)$$

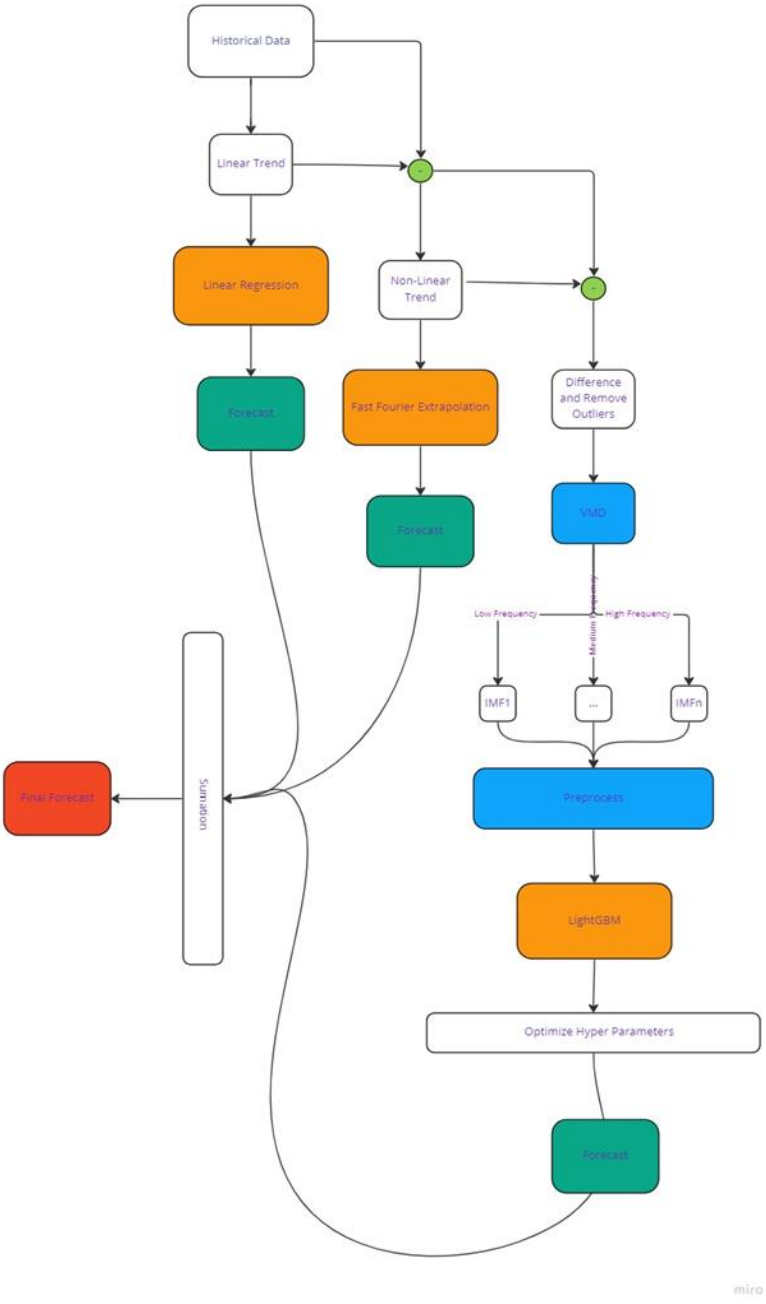
دو پارامتر α, β با بهینه سازی از داده‌های تست گذشته بدست می‌آیند، قیمت کمینه هم به صورت مشابه می‌توان تخمین زد. حال شرط لازم برای معامله گری را نسبت حد سود به ضرر تعین می‌کند که مقدار بهینه آن یک تنظیم شده است

```
def Strategy(df, info, param):

    """
    This function takes a pandas DataFrame containing OHLC (Open, High, Low, Close)
    data for a financial instrument, a dictionary info containing information about the Forecast,
    and a dictionary param containing parameters for the trade strategy.

    The function returns a dictionary containing information about the trade strategy,
    including the take profit, step loss, RR, position size, action (buy, sell, or hold) and etc.

    """
```

در نهایت پوزیشن بدست آمده از تابع Strategy توسط تابع Control Position باز می‌شود و پس از گذشت حداکثر زمان مجاز در صورت باز بودن، بسته می‌شود.

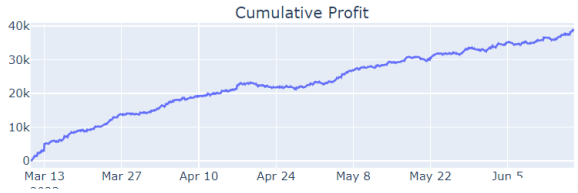
```python
def Control_Position(initialize,  trade_info, max_pending_time=2*60, max_open_time=20*60):

    """
    Control the lifecycle of a position in MetaTrader 5.

    initialize: list, contains login, password, and server information to connect to the MT5 terminal
    trade_info: dict, contains information for the trade to open, including currency pair, trade direction,
    lot size, stop loss, and take profit
    max_pending_time: int, the maximum time in seconds to wait for a pending order to execute
    max_open_time: int, the maximum time in seconds to keep an open trade before closing it

    """



    # Initialization
    mt5.initialize()
    mt5.login(login=initialize[0],password=initialize[1],server=initialize[2])

    # Open Position
    trade, request=Open_Position(trade_info)

    if request["action"]==mt5.TRADE_ACTION_PENDING:
        t1 = threading.Thread(target=Close_Position, args=(trade.order, request, 'Remove', trade_info['Currency'], max_pending_time))
        t1.start()

    t1 = threading.Thread(target=Close_Position, args=(trade.order, request,'Close', trade_info['Currency'], max_open_time))
    t1.start()
```

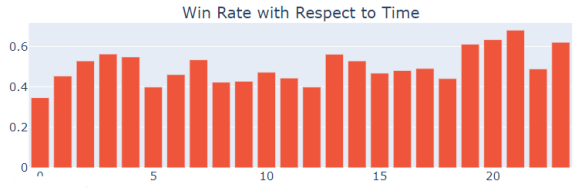
چارچوب کلی روش پیشنهادی

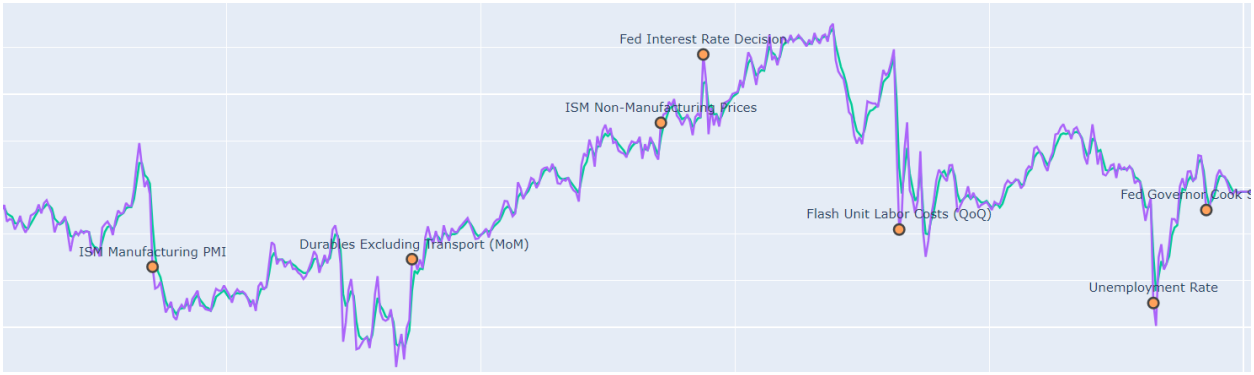


برای بررسی عملکرد مدل quant، به مدت سه ماه آن را بر روی داده‌های گذشته جفت ارز یورو دلار در تایم فریم ۱۵ دقیقه تست گرفته‌ایم که نتایج آن به شرح زیر است:

- **Total Profit:** 38647.19182201856
- **Win Rate:** 0.4945770065075922
- **Connectivity Loss**: 11
- **Number of Trade**: 1383
- **Max Profit:** 2404.1985804248666
- **Min Profit:** -100.0
- **Mean Profit:** 27.94446263341906
- **MDD**: 1.8156830182905583







طبق نتایج فوق، می‌توان موارد زیر را برداشت کرد:

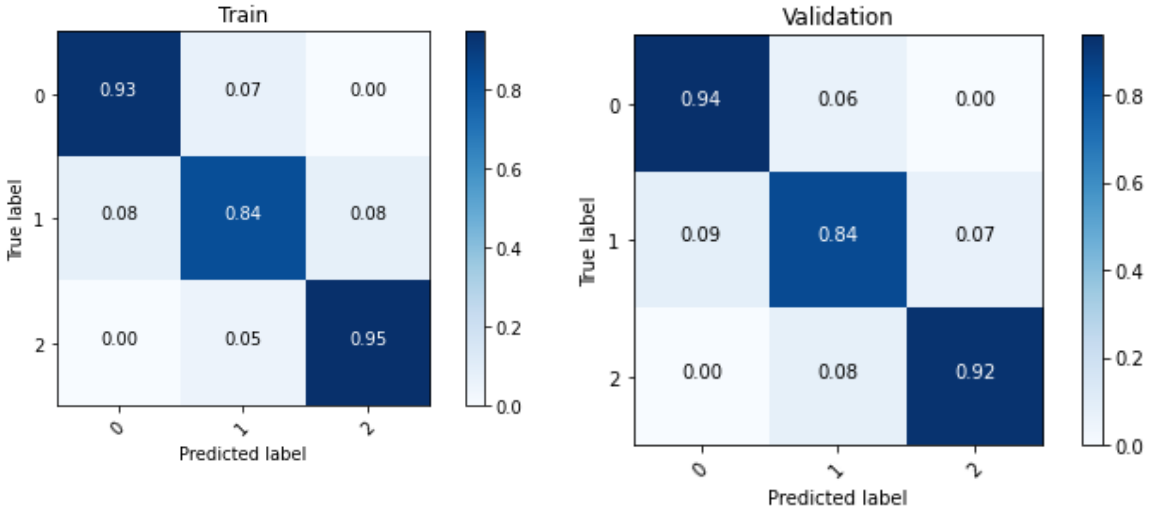
- ✓ از مجموع ۱۰ هزار پیش‌بینی تنها ۱۳۸۳ از آنها شرط لازم برای ورود به معامله را داشتند که ۵۰ درصد از آنها سود ده بوده و جمعاً ۳۸۶۴۷ دلار سود به ازای ریسک ۱۰۰ دلار در هر معامله بدست آمده است. بیشتر ضرر و کاهش سرمایه برابر ۱,۸ درصد بوده و به صورت متوسط هر معامله مقدار ۲۷ دلار سود ده بوده است.
- ✓ در تمامی ساعت روز عملکرد مدل تقریبا یکسان بوده اما ساعت ۷،۸،۹ شب به تایم متا تریدر، بهترین ساعت و ساعت ۱۲ بدترین (شکل قرمز رنگ)
- ✓ به طور کلی بیشترین خطا مدل مربوط به تایم های اخبار مهم آمریکا بوده و زمان هایی که بازار روندی رنج را داشته است

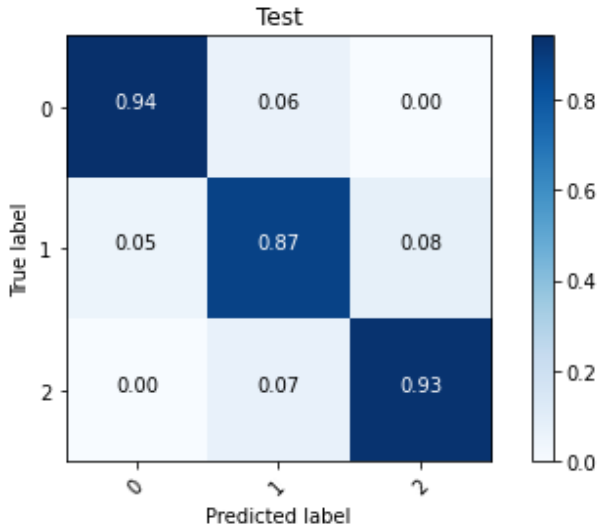
Advance quant Model

برای بهبود کارایی مدل quant ، مدل advance quant مطرح شده که عملکرد آن به شرح زیر است:
ابتدا مدل quant classifier اجرا می‌شود و جهت حرکت قیمت را پیش بینی می‌کند. این پیش بینی می‌تواند از جنس خرید، فروش و یا خنثی باشد. در صورتی که پیش بینی برابر خنثی نباشد، مدل quant اجرا می‌شود. به این ترتیب از معامله گردی در زمان هایی که بازار روند رنج داشته باشد جلوگیری می‌شود.
مدل quant classifier در حقیقت یک Gated Transformer است که از تاخیرهای مدهای تجزیه شده با الگوریتم VMD استفاده می‌کند.

```
forecast, prob, net, acc, encoding, label_indices=transformer_model(inputs, labels, inputs_forecast,
                                                    save_name=save_name, use_pre_train=use_pre_train,
                                                    hyper_tune=hyper_tune, plotResults=plot_results, n_trials=n_trials)
```

Confusion Matrix این مدل برای جفت ارز یورو دلار در تایم فریم ۱۵ دقیقه به صورت زیر است:





علاوه بر آن مدل می‌تواند با تبدیل کندل‌ها به کندل های هیکن آشی، که رفتاری نرم تر را نشان می‌دهند، روند بازار را نیز پیش‌بینی کند.

```
if use_haiken_ashi:
    y=label_returns(find_heikin_ashi_candlestick(df))
else:
    y=label_returns(df)
```

پس از پیش‌بینی جهت قیمت (صعودی، نزولی، خنثی) در صورت خنثی نبودن، اطلاعات لایه اخر مدل به یک شبکه MLP داده می‌شود تا مقدار میانگین قیمت را پیش بینی کند. بقیه مراحل دقیقا مانند مدل quant انجام می‌شود.

```
if forecast1!=1:

    df=correct_candle(initialize, currency, TimeFrame)
    targets=np.diff(df['Mean'].values)[-len(encoding)+1:]
    y_scaled,scaler=standardization(targets.reshape(-1, 1))
    encoding = np.nan_to_num(encoding)
    forecast, mlp, rmse=mlp_regressor(encoding[:-1,:], y_scaled, encoding[-1,:], f'{currency}_{TimeFrame}',
                hyper_tune=hyper_tune, plotResults=plot_results, n_trials=30)
```