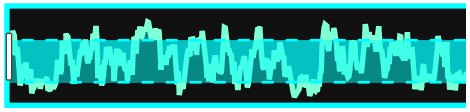
Crypto Price Interactive Dashboard(Daily Update)

```
START_DATE = "2024-01-11" # start date for historical data
RSI_TIME_WINDOW = 7 #number of days


import requests #библиотека для выполнения HTTP-запросов.
import pandas as pd #мощная библиотека для работы с данными, таблицами и временными рядами.
import warnings # модуль стандартной библиотеки Python для управления предупреждениями.
import datetime as dt #модуль для работы с датами и временем.
import plotly.express as px #высокоуровневый интерфейс Plotly для быстрого построения графиков.
import matplotlib.pyplot as plt # основной интерфейс для построения статических графиков в Matplotlib.
import plotly.graph_objects as go  # импорт графических объектов Plotly
import pandas_datareader.data as web  # доступ к данным через pandas-datareader
from plotly.subplots import make_subplots  # утилита для создания нескольких графиков в одной фигуре
warnings.filterwarnings('ignore')  # подавление предупреждений (чтобы шум не засорял вывод)

## URLS and names
urls = ["https://www.cryptodatadownload.com/cdd/Bitfinex_EOSUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_EDOUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_BTCUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_ETHUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_LTCUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_BATUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_OMGUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_DAIUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_ETCUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_ETPUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_NEOUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_REPUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_TRXUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_XLMUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_XMRUSD_d.csv",
        "https://www.cryptodatadownload.com/cdd/Bitfinex_XVGUSD_d.csv",
       ]
crypto_names = ["EOS Coin (EOS)",
                "Eidoo (EDO)",
                "Bitcoin (BTC)",
                "Ethereum (ETH)",
                "Litecoin (LTC)",
                "Basic Attention Token (BAT)",
                "OmiseGO (OMG)",
                "Dai (DAI)",
                "Ethereum Classic (ETC)",
                "Metaverse (ETP)",
                "Neo (NEO)",
                "Augur (REP)",
                "TRON (TRX)",
                "Stellar (XLM)",
                "Monero (XMR)",
                "Verge (XVG)"
               ]

## Data download and loading
def df_loader(urls , start_date = "2021-01-01"):
## Функция загрузки и обработки данных
def df_loader(urls , start_date = "2021-01-01"):
    filenames = []
    all_df = pd.DataFrame()
    for idx,url in enumerate(urls):
        # Загрузка CSV файлов
        req = requests.get(url,verify=False)
        url_content = req.content
        filename = url[48:]  # извлечение имени файла из URL
        csv_file = open( filename , 'wb')
        csv_file.write(url_content)
        csv_file.close()
        filename = filename[:-9]  # удаление расширения
        filenames.append(filename)
    # Обработка каждого файла
    for file in filenames:
        df = pd.read_csv(file + "USD_d.csv", header = 1, parse_dates=["date"])
        df = df [df["date"] > start_date]  # фильтрация по дате
        df.index = df.date  # установка даты как индекса
        # Удаление ненужных колонок
        df.drop(labels = [df.columns[0],df.columns[1],df.columns[8]] , axis = 1 , inplace = True)
        all_df = pd.concat([all_df,df], ignore_index=False)  # объединение данных

    return all_df , filenames
```

```
# Функция расчета RSI (Relative Strength Index)
def computeRSI (data, time_window):
    diff = data.diff(1).dropna()  # разница цен
    up_chg = 0 * diff  # положительные изменения
    down_chg = 0 * diff  # отрицательные изменения
    up_chg[diff > 0] = diff[ diff>0 ]
    down_chg[diff < 0] = diff[ diff < 0 ]
    # Экспоненциальное скользящее среднее
    up_chg_avg = up_chg.ewm(com=time_window-1, min_periods=time_window).mean()
    down_chg_avg = down_chg.ewm(com=time_window-1, min_periods=time_window).mean()
    rs = abs(up_chg_avg/down_chg_avg)  # относительная сила
    rsi = 100 - 100/(1+rs)  # индекс относительной силы
    return rsi

# Загрузка данных
all_df , filenames = df_loader(urls , start_date=START_DATE)

# Обработка каждой криптовалюты
crypto_df = []
for file in filenames:
    symbol = file + str("/USD")
    temp_df = pd.DataFrame(all_df[all_df["symbol"] == symbol])  # фильтрация по символу
    temp_df.drop(columns= ["symbol"] ,inplace = True)  # удаление колонки symbol
    temp_df["close_rsi"] = computeRSI(temp_df['close'], time_window=RSI_TIME_WINDOW)  # расчет RSI
    temp_df["high_rsi"] = 30  # уровень перепроданности
    temp_df["low_rsi"] = 70   # уровень перекупленности
    exec('%s = temp_df.copy()' % file.lower())  # создание переменной с именем криптовалюты
    crypto_df.append(temp_df)  # добавление в список

## Создание графика
fig = make_subplots(rows=3, cols=2, shared_xaxes=True, specs=[[{"rowspan": 2}, {"rowspan": 2}], [{"rowspan": 1}, {"rowspan":

# Кнопки для выбора временного диапазона
date_buttons = [ ... ]

# Кнопки для выбора криптовалюты
buttons = [ ... ]

# Добавление графиков для каждой криптовалюты
for df in crypto_df:
    # Свечной график
    fig.add_trace(go.Candlestick(...), row=1, col=1)
    # Объемы торгов
    fig.add_trace(go.Bar(...), row=3, col=1)
    # Линейный график цены
    fig.add_trace(go.Scatter(...), row=1, col=2)
    # Зоны high/low
    fig.add_trace(go.Scatter(...), row=1, col=2)
    fig.add_trace(go.Scatter(...), row=1, col=2)
    # График RSI
    fig.add_trace(go.Scatter(...), row=3, col=2)
    # Зоны RSI 30/70
    fig.add_trace(go.Scatter(...), row=3, col=2)
    fig.add_trace(go.Scatter(...), row=3, col=2)

# Настройка осей и внешнего вида
fig.update_xaxes(...)
fig.update_layout(...)
fig.update_yaxes(...)

# Настройка видимости графиков (изначально скрыты)
for i in range(0,16*COUNT):
    fig.data[i].visible = False
for i in range(COUNT):
    fig.data[i].visible = True

# Дополнительные настройки интерфейса
fig.layout["xaxis"]["rangeslider"]["visible"] = False
# ... другие настройки

# Отображение графика
fig.show()
```

- Click the 🏠 icon on the RIGHT TOP to reset the dashboard.
- Start date for historical data is 1st January 2024.
- Total 16 cryptocurrencies are used in the dashboard. Check them here.
- Relative Strength Index (RSI) has a period of 7 days. Find more about RSI here.



If you find this notebook useful, support with an upvote👍

Этот код на Python выполняет загрузку, обработку и визуализацию данных о криптовалютах. Вот основные его компоненты и функции:

1. Определение параметров

START_DATE: Начальная дата для загрузки исторических данных (11 января 2024 года). RSI_TIME_WINDOW: Период для расчета индикатора относительной силы (RSI), установленный на 7 дней.

2. Импорт библиотек

Код использует несколько библиотек:

requests: Для загрузки данных из интернета. pandas: Для обработки и анализа данных. plotly: Для создания интерактивных графиков. matplotlib: Для визуализации. pandas_datareader: Для доступа к финансовым данным.

3. Загрузка данных

Функция df_loader:

Загружает данные о криптовалютах из указанных URL. Сохраняет их в CSV-файлы и объединяет в один DataFrame. Фильтрует данные по дате. 4. Расчет RSI Функция computeRSI:

Рассчитывает RSI на основе закрывающих цен криптовалют. Использует экспоненциальное скользящее среднее для вычисления средних изменений. 5. Подготовка данных для графиков Создается список DataFrame для каждой криптовалюты, в который добавляется столбец с RSI и установлены уровни для низкого и высокого RSI (30 и 70 соответственно). 6. Построение графиков Используется make_subplots для создания многопанельного графика, который включает в себя: Свечные графики для цен. Графики объема торгов. Графики цен с линиями RSI. 7. Настройка графиков Добавляются кнопки для выбора временного диапазона и криптовалюты. Настраиваются оси и внешний вид графиков (цвета, размеры шрифтов и т.д.). Устанавливаются видимость ползунков диапазона для осей X. 8. Отображение графика В конце код вызывает fig.show(), чтобы отобразить интерактивный график. Заключение Этот код позволяет загружать данные о различных криптовалютах, рассчитывать их RSI и визуализировать информацию в удобном формате, что полезно для анализа рынка. Если вам нужно больше информации по какому-либо из пунктов, дайте знать!

Ниже представлена графическая блок-схема алгоритма кода

```
flowchart TD
    A[Начало] --> B[Задать параметры: <br>START_DATE, RSI_TIME_WINDOW]
    B --> C[Импортировать библиотеки <br>(requests, pandas, plotly и др.)]
    C --> D[Определить URL и названия <br>криптовалют]
    D --> E[Вызвать функцию df_loader() <br>и загрузить данные: <br>- Загрузить CSV <br>- Объединить данные <br>- Отфильтроват
    E --> F[Для каждого файла: <br>- Рассчитать RSI <br>- Создать DataFrame для каждой криптовалюты]
```

```
F --> G[Создать графики: <br>- Свечные графики <br>- Графики объемов <br>- Графики RSI]
G --> H[Настроить графики: <br>- Кнопки для выбора валюты <br>- Настройка осей <br>- Оформление]
H --> I[Отобразить график <br>с помощью fig.show()]
I --> J[Конец]
```

Как использовать Чтобы отобразить блок-схему, вам нужно воспользоваться инструментом для рендеринга диаграмм в формате Mermaid (например, в GitHub, GitLab или специальном редакторе для Mermaid).

```
"""
Рабочий пример кода по вашему flowchart'y:
- инициализация констант
- импорт библиотек
- определение URLs и названий криптовалют
- функция df_loader: загружает данные в виде DataFrame и возвращает список DataFrame-ов и_names
- функция computeRSI: вычисляет RSI для одного DataFrame
- загрузка данных и вычисление RSI для нескольких монет
"""

from __future__ import annotations

import pandas as pd
import numpy as np
import requests
from io import StringIO
from typing import List, Tuple

# --------------------------------
# Константы
START_DATE = "2020-01-01"
RSI_TIME_WINDOW = 14

# --------------------------------
# Константы: названия сайтов/URL-адресов и тикеры монет
# Пример: словарь <название> -> <URL>. Замените на ваши источники данных.
COINS = [
    {"name": "Bitcoin", "symbol": "BTC", "url": "https://example.com/btc.csv"},
    {"name": "Ethereum", "symbol": "ETH", "url": "https://example.com/eth.csv"},
    {"name": "Cardano", "symbol": "ADA", "url": "https://example.com/ada.csv"},
    # добавьте нужные монеты
]

# --------------------------------
# Функции

def df_loader(urls: List[str], date_start: str | None = None) -> Tuple[List[pd.DataFrame], List[str]]:
    """
    Загружает CSV-файлы по списку URL и возвращает:
    - список DataFrame'ов (по каждой монете)
    - список названий файлов/coin'ов (для сопоставления)

    Примечание: в данном примере предполагается, что CSV имеет колонки
    Date, Open, High, Low, Close, Volume (или аналогичные).
    """
    dataframes: List[pd.DataFrame] = []
    filenames: List[str] = []

    for c in COINS:
        url = c["url"]
        name = c["name"]
        filenames.append(name)

        try:
            # Попытка скачать CSV
            resp = requests.get(url, timeout=15)
            resp.raise_for_status()
            content = resp.text

            # Предположим, CSV имеет заголовки и колонку "Date" и "Close"
            df = pd.read_csv(StringIO(content), parse_dates=["Date"])
            if date_start:
                df = df[df["Date"] >= pd.to_datetime(date_start)]
            df = df.sort_values("Date").reset_index(drop=True)

            # Проверка минимальных столбцов
            required_cols = {"Date", "Close"}
            if not required_cols.issubset(set(df.columns)):
                raise ValueError(f"CSV для {name} не содержит необходимых столбцов.")

            dataframes.append(df)
        except Exception as e:
```

```
            print(f"Ошибка загрузки данных для {name} по {url}: {e}")
            dataframes.append(pd.DataFrame(columns=["Date", "Close"]))  # пустой/DataFrame-заглушка

    return dataframes, filenames


def computeRSI(series: pd.Series, window: int = RSI_TIME_WINDOW) -> pd.Series:
    """
    Вычисление RSI по серии цен Close.
    RSI = 100 - (100 / (1 + RS))
    где RS = средняя доходность/средняя потеря за окно.
    """
    if len(series) < window:
        return pd.Series([np.nan] * len(series), index=series.index)

    delta = series.diff().astype(float)
    up = delta.clip(lower=0)
    down = -delta.clip(upper=0)

    # Скользящие средние
    roll_up = up.rolling(window=window, min_periods=window).mean()
    roll_down = down.rolling(window=window, min_periods=window).mean()

    rs = roll_up / roll_down
    rsi = 100.0 - (100.0 / (1.0 + rs))

    return rsi

# ------------------------------
# Основной сценарий

def main():
    # 1) Инициализация констант (уже сделано выше)

    # 3) Определение URLs и названий криптовалют (уже сделано выше)
    # 4) Функции df_loader и computeRSI реализованы

    # 5) Загрузка данных
    all_dfs, names = df_loader([c["url"] for c in COINS], date_start=START_DATE)

    # 6) Вычисление RSI для каждого датафрейма
    results = {}
    for name, df in zip(names, all_dfs):
        if df.empty:
            results[name] = pd.Series(dtype=float)
            continue
        if "Close" not in df.columns:
            results[name] = pd.Series(dtype=float)
            continue
        rsi = computeRSI(df["Close"], window=RSI_TIME_WINDOW)
        results[name] = rsi

    # Пример: вывести последние значения RSI для каждой монеты
    for name, rsi_series in results.items():
        if len(rsi_series) == 0:
            print(f"{name}: нет данных")
        else:
            last = rsi_series.dropna().iloc[-1]
            print(f"{name}: последний RSI = {last:.2f}")

if __name__ == "__main__":
    main()
```

```
Ошибка загрузки данных для Bitcoin по https://example.com/btc.csv: 404 Client Error: Not Found for url: https://example.com/
Ошибка загрузки данных для Ethereum по https://example.com/eth.csv: 404 Client Error: Not Found for url: https://example.com
Ошибка загрузки данных для Cardano по https://example.com/ada.csv: 404 Client Error: Not Found for url: https://example.com/
Bitcoin: нет данных
Ethereum: нет данных
Cardano: нет данных
```

```
%%html
<!--[if IE]><meta http-equiv="X-UA-Compatible" content="IE=5,IE=9" ><![endif]-->
<!DOCTYPE html>
<html>
<head>
<title>Диаграмма без названия</title>
<meta charset="utf-8"/>
</head>
<body><div class="mxgraph" style="max-width:100%;border:1px solid transparent;" data-mxgraph="{&quot;highlight&quot;:&quot;
<script type="text/javascript" src="https://viewer.diagrams.net/js/viewer-static.min.js"></script>
</body>
</html>
```