

بسم الله الرحمن الرحيم



سیستم تحلیل مالی و معاملات بلادرنگ با هوش مصنوعی

پروژه نهایی درس تحلیل‌ها و سیستم‌های داده حجیم



استاد درس: دکتر حسن نادری

دانشجویان: علی احمدی، اهورا امینی، مهدی شکاری و امیر کریمی

بهمن ۱۴۰۳

فهرست مطالب

3.....	مقدمه
3.....	هدف پروژه
3.....	اهمیت پروژه
3.....	چالش‌ها
4.....	معماری سیستم
5.....	API Nobitex
5.....	Data Ingestion Service
5.....	Apache Kafka
5.....	Stream Data Processing Service (Apache Spark)
6.....	محاسبه شاخص‌ها
6.....	تولید سیگنال‌ها
7.....	پردازش میکروبیج‌ها
7.....	خواندن و نوشتن با Spark
7.....	مدیریت استریم
7.....	سرویس میانی Kafka-to-QuestDB
7.....	پردازش پیام‌ها
7.....	جایگزینی nan و تبدیل مقادیر
8.....	ساخت قالب ILP و ارسال داده‌ها به QuestDB
8.....	QuestDB
9.....	Aggregation Service
9.....	Aggregate API
9.....	Summarize API
10.....	Summarize Multiple API
10.....	Visualization Service (Grafana)
12.....	Trending Prediction Service
12.....	وارد کردن کتابخانه‌های مورد نیاز
12.....	تابع ساخت مدل LSTM
12.....	تابع دانلود داده‌های سهام
13.....	تابع آماده سازی داده‌ها (prepare_data)
13.....	تابع بازگرداندن داده‌ها به مقیاس اصلی (inverse_transform)

13	تابع پیشبینی آینده (predict_future).....
13	تابع رسم نمودار (plot_dynamic_symbol).....
13	حلقه اصلی اجرای کد (__main__).....
14	تکنولوژی‌های استفاده شده.....
14	API Nobitex.....
14	نحوه دریافت داده.....
14	چرا این API انتخاب شده است؟.....
14	Apache Kafka.....
14	چرا Kafka انتخاب شده است؟.....
14	معماری Kafka.....
15	Apache Spark.....
15	چرا اسپارک انتخاب شده است؟.....
16	معماری Spark.....
16	QuestDB.....
16	چرا از QuestDB استفاده شده است؟.....
17	Minikube.....
17	چرا Minikube انتخاب شده است؟.....
18	الگوریتم LSTM.....
18	ساختار LSTM.....
18	چرا LSTM برای تحلیل داده‌های مالی مناسب است؟.....
18	چالش‌های استفاده از LSTM در تحلیل مالی.....
18	راهکارهای افزایش دقت مدل.....
19	الگوریتم‌های معاملاتی (اندیکاتورها).....
19	میانگین متحرک ساده (SMA).....
19	میانگین متحرک نمایی (EMA).....
19	شاخص قدرت نسبی (RSI).....
20	نتیجه‌گیری.....

مقدمه

هدف پروژه

پروژه تحلیل داده‌های مالی در بازارهای مالی مانند رمزارز و نمادهای سهام مختلف به منظور تجزیه و تحلیل داده‌ها در زمان واقعی (Real-time) طراحی و پیاده‌سازی شده است. این سیستم با استفاده از تکنولوژی‌های توزیع شده و پردازش بلادرنگ داده‌ها، امکان تحلیل سریع و تصمیم‌گیری به موقع برای معامله‌گران بازارهای مالی را فراهم می‌کند.

بازارهای مالی به دلیل نوسانات شدید و لحظه‌ای قیمت‌ها، نیاز به ابزارهایی دارند که بتوانند به سرعت و به صورت دقیق، اطلاعات را پردازش کرده و سیگنال‌های مناسب برای خرید و فروش ارائه دهند. در این پروژه، هدف اصلی این است که با استفاده از پردازش‌های پیچیده و بلادرنگ، سیگنال‌های دقیق و به موقع برای معامله‌گران فراهم شود تا بتوانند تصمیمات بهتری اتخاذ نمایند.

اهمیت پروژه

تحلیل داده‌های مالی از اهمیت بسیار بالایی برخوردار است. با افزایش حجم داده‌ها و نیاز به پردازش بلادرنگ، استفاده از سیستم‌های مقیاس‌پذیر و توزیع شده مانند Apache Spark، Apache Kafka و QuestDB به ویژه در تحلیل‌های مالی اهمیت پیدا می‌کند. این پروژه نه تنها بر مقیاس‌پذیری تمرکز دارد، بلکه توانایی پردازش داده‌ها در لحظه را نیز فراهم می‌آورد.

به طور ساده، این پروژه یک سیستم بلادرنگ است که به صورت خودکار، داده‌ها را از بازارهای مالی دریافت کرده، پردازش کرده و سپس سیگنال‌های مناسب خرید و فروش و همچنین با استفاده از هوش مصنوعی پیشبینی روند را نیز به کاربران نمایش می‌دهد.

این توانایی می‌تواند برای کسانی که در بازارهای مالی فعالیت می‌کنند، به ویژه برای کسانی که به دنبال تصمیم‌گیری‌های سریع و بهینه هستند، بسیار ارزشمند باشد.

چالش‌ها

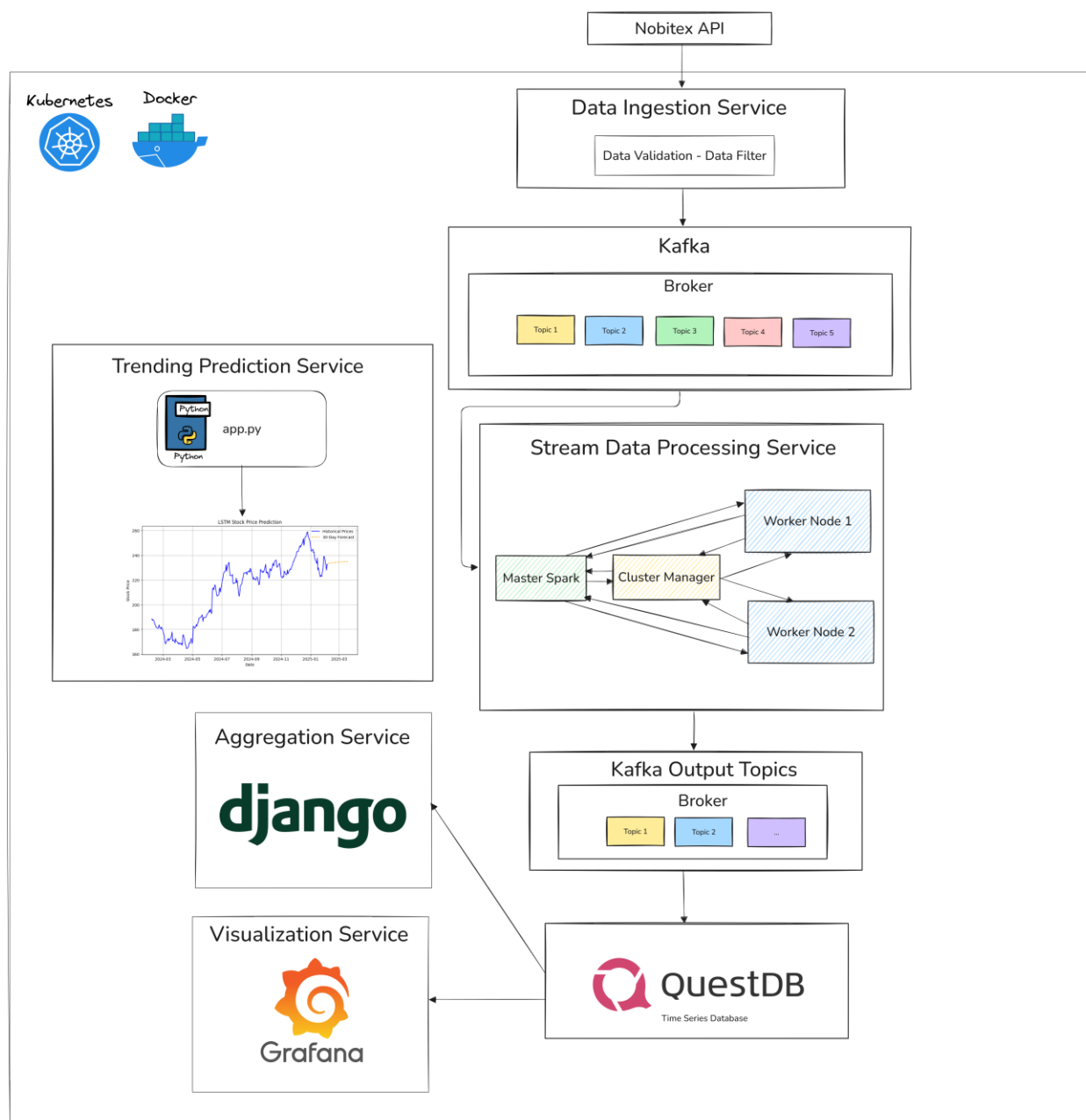
در این پروژه، برخی از چالش‌ها به شرح زیر هستند:

- **حجم بالای داده‌ها:** با توجه به حجم عظیم داده‌های موجود در بازارهای مالی، پردازش سریع داده‌ها به صورت بلادرنگ نیازمند استفاده از سیستم‌های توزیع شده است.
- **تاخیر در پردازش (Latency):** سیستم باید قادر باشد تا داده‌ها را با کمترین تاخیر پردازش کرده و سیگنال‌های دقیق و به موقع برای کاربران ارسال نماید.
- **مقیاس‌پذیری:** با توجه به اینکه داده‌ها به صورت لحظه‌ای وارد سیستم می‌شوند، مقیاس‌پذیری سیستم برای مدیریت تعداد زیاد درخواست‌ها و داده‌ها ضروری است.
- **پیشبینی روند با استفاده از هوش مصنوعی:** به دلیل پیچیدگی ذاتی داده‌های مالی و همچنین پیشبینی آن‌ها، استفاده از الگوریتم مناسب برای تحلیل روند در آینده بسیار چالش برانگیز و محل بحث است.

هدف از این پروژه این است که با استفاده از معماری‌های مقیاس‌پذیر، پردازش‌های توزیع شده و استفاده از هوش مصنوعی (یادگیری عمیق) این چالش‌ها را پشت سر گذاشته و سیستمی کارآمد و سریع برای تحلیل داده‌های بازارهای مالی ارائه دهد.

معماری سیستم

معماری سیستم به شرح زیر است:



این معماری یک سیستم توزیع شده برای پردازش داده‌ها به صورت بلادرنگ است که شامل اجزای مختلفی می‌باشد. در ادامه توضیحات معماری ارائه شده به تفصیل شرح داده خواهد شد.

API Nobitex

این API به عنوان منبع داده‌های رمز ارز عمل می‌کند. داده‌ها شامل اطلاعات معاملات، قیمت‌ها، حجم بازار و سایر معیارهای مالی مرتبط با رمزارزها می‌باشد. ساختار داده‌های دریافتی به شرح زیر است:

نام متغیر	نوع متغیر	توضیحات
stock_symbol	object	نماد سهام
local_time	timestamps	تاریخ و ساعت دقیق دریافت داده
open_prices	float	قیمت باز شدن (بر اساس کندل)
high_prices	float	بالاترین قیمت
low_prices	float	پایین‌ترین قیمت
close_prices	float	قیمت بسته شدن
volumes	int	حجم معامله بازار

Data Ingestion Service

وظیفه این سرویس جمع‌آوری و انتقال داده از API Nobitex است. در این فرآیند، داده‌ها مورد اعتبارسنجی (Validation) و فیلترگذاری قرار می‌گیرند تا فقط داده‌های معتبر به مرحله بعدی ارسال شوند. در صورت صحت داده‌ها، توسط کد producer هر داده جدید بر اساس فیلد stock_symbol به یک Topic در Kafka ارسال می‌شود.

برای مثال داده‌های مربوط به BTCIRT به تاپیک btcirt_topic ارسال خواهد شد.

```
2025-01-25 18:35:44 INFO:root:Data sent to topic=shibirt_topic, partition=0, offset=7, data={'stock_symbol': 'SHIBIRT', 'local_time': '2025-01-25 18:34:00', 'open': 1680.2, 'high': 1684.6, 'low': 1680.1, 'close': 1680.1, 'volume': 811.0, 'topic': 'shibirt_topic'}
2025-01-25 18:35:44 INFO:root:payload: {'stock_symbol': 'SHIBIRT', 'local_time': '2025-01-25 18:34:00', 'open': 1680.2, 'high': 1684.6, 'low': 1680.1, 'close': 1680.1, 'volume': 811.0, 'topic': 'shibirt_topic'} processed and forwarded to Kafka topic: shibirt_topic
2025-01-25 18:36:45 INFO:root:Data sent to topic=btctrt_topic, partition=0, offset=7, data={'stock_symbol': 'BTCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 8697888889.0, 'high': 8697888889.0, 'low': 8697888889.0, 'close': 8697888889.0, 'volume': 5.22939e-05, 'topic': 'btctrt_topic'}
2025-01-25 18:36:45 INFO:root:payload: {'stock_symbol': 'BTCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 8697888889.0, 'high': 8697888889.0, 'low': 8697888889.0, 'close': 8697888889.0, 'volume': 5.22939e-05, 'topic': 'btctrt_topic'} processed and forwarded to Kafka topic: btctrt_topic
2025-01-25 18:36:53 INFO:root:Data sent to topic=usdtirt_topic, partition=0, offset=8, data={'stock_symbol': 'USDIRT', 'local_time': '2025-01-25 18:36:00', 'open': 83597.0, 'high': 83598.0, 'low': 83552.0, 'close': 83553.0, 'volume': 2718.0373713987, 'topic': 'usdtirt_topic'}
2025-01-25 18:36:53 INFO:root:payload: {'stock_symbol': 'USDIRT', 'local_time': '2025-01-25 18:36:00', 'open': 83597.0, 'high': 83598.0, 'low': 83552.0, 'close': 83553.0, 'volume': 2718.0373713987, 'topic': 'usdtirt_topic'} processed and forwarded to Kafka topic: usdtirt_topic
2025-01-25 18:36:54 INFO:root:Data sent to topic=ethirt_topic, partition=0, offset=8, data={'stock_symbol': 'ETHIRT', 'local_time': '2025-01-25 18:36:00', 'open': 277734600.0, 'high': 277734600.0, 'low': 277170000.0, 'close': 277734600.0, 'volume': 0.2178, 'topic': 'ethirt_topic'}
2025-01-25 18:36:54 INFO:root:payload: {'stock_symbol': 'ETHIRT', 'local_time': '2025-01-25 18:36:00', 'open': 277734600.0, 'high': 277734600.0, 'low': 277170000.0, 'close': 277734600.0, 'volume': 0.2178, 'topic': 'ethirt_topic'} processed and forwarded to Kafka topic: ethirt_topic
2025-01-25 18:36:57 INFO:root:Data sent to topic=etctrt_topic, partition=0, offset=8, data={'stock_symbol': 'ETCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 2261992.0, 'high': 2261992.0, 'low': 2261992.0, 'close': 2261992.0, 'volume': 0.0, 'topic': 'etctrt_topic'}
2025-01-25 18:36:57 INFO:root:payload: {'stock_symbol': 'ETCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 2261992.0, 'high': 2261992.0, 'low': 2261992.0, 'close': 2261992.0, 'volume': 0.0, 'topic': 'etctrt_topic'} processed and forwarded to Kafka topic: etctrt_topic
```

Apache Kafka

داده‌های تصحیح شده توسط سرویس Data Ingestion به Kafka به عنوان یک سیستم مقیاس‌پذیر و پایدار وارد می‌شوند. امکان انتقال داده‌ها به اجزای مختلف را به شیوه‌ای بسیار کارآمد فراهم می‌کند.

Broker: مسئول توزیع داده‌ها در قالب Topic‌های مختلف است. در معماری سیستم چندین Topic نمایش داده شده است که هر کدام ممکن است یک جزئی از داده را نمایش دهند.

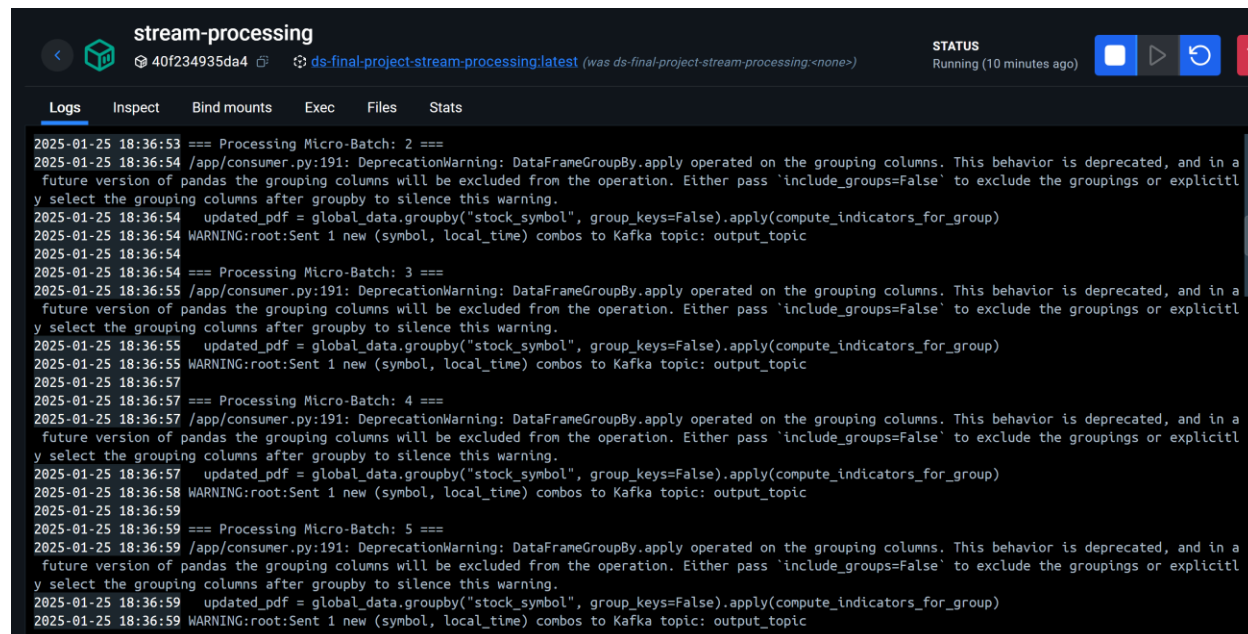
Stream Data Processing Service (Apache Spark)

داده‌های ارسال شده از Kafka در اینجا پردازش می‌شوند. Apache Spark به عنوان یک محیط پردازش موازی برای انجام عملیات پیچیده روی داده‌ها عمل می‌کند.

Cluster Spark و Master Spark: مدیریت عملیات پردازش توزیع شده را برعهده دارند که داده‌ها میان Worker Node1 و Worker Node2 تقسیم می‌شوند.

Kafka Output Topics: داده‌های پردازش شده مجدداً به Kafka ارسال می‌شوند، جایی که موضوعات خروجی جدید برای این داده‌ها ایجاد می‌شود.

این سرویس شامل چندین بخش است که به طور کلی برای پردازش و داده‌های استریم از Kafka، محاسبه شاخص‌های معاملاتی یا همان اندیکاتورها (SMA, EMA, RSI) و ارسال داده به Kafka و مدیریت میکروبیج‌ها با Spark طراحی شده است.



```
stream-processing
40f234935da4 ds-final-project-stream-processing:latest (was ds-final-project-stream-processing:<none>)
STATUS
Running (10 minutes ago)
Logs Inspect Bind mounts Exec Files Stats
2025-01-25 18:36:53 === Processing Micro-Batch: 2 ===
2025-01-25 18:36:54 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:54 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:54 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:54 === Processing Micro-Batch: 3 ===
2025-01-25 18:36:55 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:55 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:55 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:57 === Processing Micro-Batch: 4 ===
2025-01-25 18:36:57 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:57 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:58 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:59 === Processing Micro-Batch: 5 ===
2025-01-25 18:36:59 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:59 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:59 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
```

محاسبه شاخص‌ها

این بخش برای هر گروه از داده‌ها شاخص‌های زیر را محاسبه می‌کند:

- **SMA (5)**: میانگین ساده در دوره‌های ۵ دقیقه‌ای
- **EMA (10)**: میانگین نمایی دوره‌های ۱۰ دقیقه‌ای
- **RSI (10)**: شاخص قدرت نسبی در دوره‌های ۱۰ دقیقه‌ای
- **Average gain & Average Loss**: در دوره‌های ۱۰ دقیقه‌ای

تولید سیگنال‌ها

در این بخش سیگنال‌های خرید، فروش یا نگه داشتن بر اساس شرایط زیر تولید می‌شود:

- **سیگنال خرید**: اگر $EMA(10) > SMA(5)$ باشد و $RSI(10)$ نیز کمتر از ۷۰ باشد.
- **سیگنال فروش**: اگر $EMA(10) < SMA(5)$ باشد و $RSI(10)$ بیشتر از ۳۰ باشد.
- در غیر این صورت سیگنال «نگه داشتن» یا همان hold است.

پردازش میکروبیج‌ها

هر میکروبیج داده‌های زیر را پردازش می‌کند:

- تبدیل داده‌های Spark به Pandas
- محاسبه شاخص‌ها و سیگنال‌ها
- فیلتر کردن داده‌هایی که قبلاً ارسال شده‌اند.
- ارسال داده‌های جدید به Kafka

خواندن و نوشتن با Spark

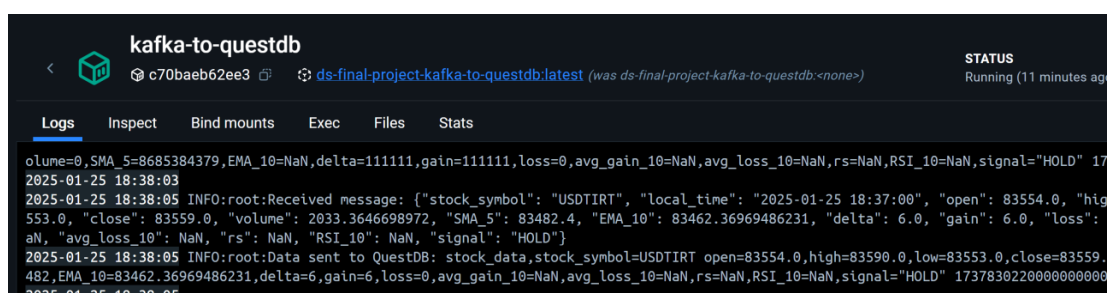
- داده‌ها از Kafka با فرمت Json خوانده می‌شوند و با استفاده از foreachBatch به صورت دسته‌ای پردازش می‌شوند.
- یک اسکیمای مشخص برای ستون‌های داده تعریف شده است.

مدیریت استریم

داده‌ها به صورت استریم از Kafka خوانده شده، شاخص‌ها محاسبه شده و داده‌های نهایی مجدداً به output_topic ارسال می‌شوند.

سرویس میانی Kafka-to-QuestDB

این سرویس برای دریافت داده‌ها از Kafka، پردازش آن‌ها و ارسال آن‌ها به QuestDB از طریق Influx Line Protocol یا ILP طراحی شده است. در ادامه بخش‌های مختلف آن توضیح داده می‌شود.



The screenshot shows the 'kafka-to-questdb' application interface. At the top, there's a header with the application name, a status bar indicating it's 'Running (11 minutes ago)', and navigation tabs for 'Logs', 'Inspect', 'Bind mounts', 'Exec', 'Files', and 'Stats'. The 'Logs' tab is active, displaying a log stream. The logs show messages received from Kafka, including stock data for 'USD TIRT' with fields like 'open', 'high', 'low', 'close', 'volume', 'SMA_5', 'EMA_10', 'delta', 'gain', 'loss', 'avg_gain_10', 'avg_loss_10', 'rs', 'RSI_10', and 'signal'. The signal is currently 'HOLD'.

- **Kafka Consumer**: برای دریافت پیام‌ها از Kafka
- **Socket**: برای ارتباط با QuestDB از طریق پروتکل ILP
- **کتابخانه‌های Pandas و math**: برای پردازش داده‌ها

پردازش پیام‌ها

- پیام‌ها از Kafka دریافت و مقدار آن‌ها (JSON) پردازش می‌شود.
- اگر پیام شامل داده‌های نامعتبر باشد، خطا گزارش می‌شود.

جایگزینی nan و تبدیل مقادیر

- مقدار "NaN" با مقدار float('nan') جایگزین می‌شود.
- مقادیر عددی مانند volume و سایر شاخص‌ها به نوع مناسب تبدیل می‌شوند.

ساخت قالب ILP و ارسال داده‌ها به QuestDB

- داده‌ها در قالب Influx Line Protocol برای QuestDB قالب‌بندی می‌شوند.
- هر رکورد شامل اطلاعاتی مثل قیمت‌ها (open, close,...) که بیشتر در ابتدا آمده بود و شاخص‌های محاسبه شده است.

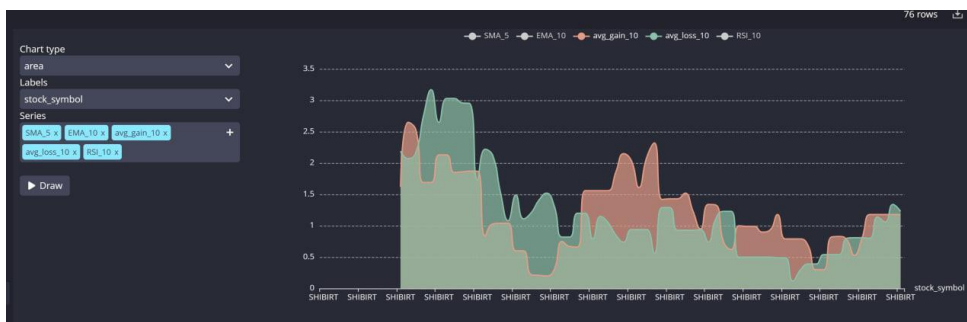
QuestDB

داده‌های پردازش شده در QuestDB ذخیره می‌شوند. QuestDB یک پایگاه داده سری زمانی است که برای انجام تحلیل‌های پیچیده و بلادرنگ استفاده می‌شوند. این پایگاه داده برای داده‌های بازارهای مالی نیز بسیار مناسب است و stock_symbol را به صورت خاص index گذاری می‌کند که به صورت سریع بتواند داده‌های مربوط به هر نماد سهام را در کوئری‌ها برگرداند.

در زیر تصویری از محیط دیتابیس نمایش داده شده است:

stock_symbol	signal	local_time	open	close	high	low	volume	SMA_5	EMA_10	delta
SHIBRT	SELL	2025-01-25T17:37:00.000000Z	1680	1680	1680	1680	91	1677	1678.679698218301	...
BTCTRT	SELL	2025-01-25T17:38:00.000000Z	8711138043	8711138042	8711138043	8711138042	0	8711136042	8711378211.468225	...
USDTIRT	SELL	2025-01-25T17:38:00.000000Z	83766	83695	83788	83695	809	83736	83768.10839324121	...
ETHIRT	BUY	2025-01-25T17:39:00.000000Z	278199995	278199995	278199995	278199995	0	278098468	278041127.34520824	...
ETCIRT	BUY	2025-01-25T17:39:00.000000Z	2255017	2255017	2255017	2255017	0	2257809	2257796.158471043	...
SHIBRT	SELL	2025-01-25T17:39:00.000000Z	1673.9	1673.9	1673.9	1673.9	0	1677	1677.81066217861	...
BTCTRT	HOLD	2025-01-25T17:40:00.000000Z	8711138043	8711138043	8711138043	8711138043	0	8711137442	8713782544.474003	...
USDTIRT	SELL	2025-01-25T17:40:00.000000Z	83712	83792	83792	83712	322	83736	83765.98686719735	...
ETHIRT	BUY	2025-01-25T17:40:00.000000Z	278199000	278199000	278199000	278199000	0	278198127	278069831.46425474	...
ETCIRT	BUY	2025-01-25T17:40:00.000000Z	2252010	2252010	2252010	2252010	0	2257207	2256744.1296581263	...
SHIBRT	SELL	2025-01-25T17:40:00.000000Z	1675.1	1675.1	1675.1	1675.1	59	1676	1677.317814509771	...

در زیر تصویری از چارت‌های QuestDB نمایش داده شده است:

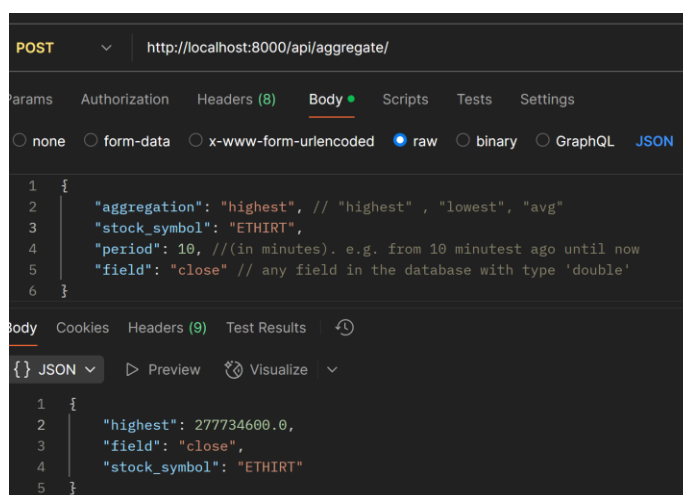


Aggregation Service

در این سرویس از یک اپلیکیشن جنگو استفاده شده است که شامل ۳ نوع API است.

Aggregate API

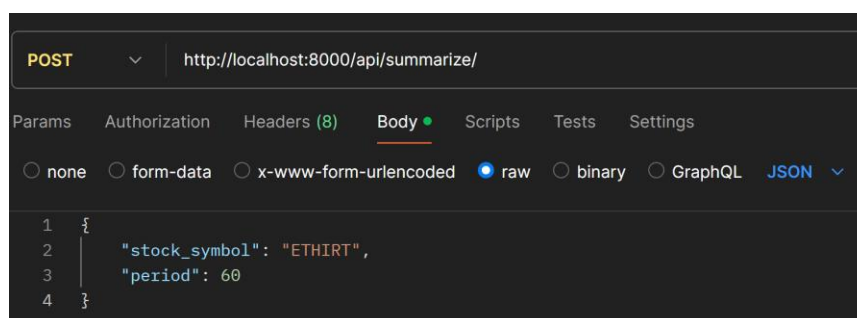
در این API با نوشتن اسم سهام مورد نظر، نوع aggregation شامل میانگین‌گیری، بیشترین یا کمترین مقدار، بازه زمانی مورد نظر و فیلد انتخابی مورد نظر جواب مناسب خواهید گرفت. برای مثال:



Summarize API

خلاصه عملکرد یک سهام در بازه زمانی مشخص را برای معیارهای SMA (5), EMA (10), RSI (10), loss, gain, close تولید می‌کند.

برای مثال ورودی:



و خروجی آن:

```
4      "summary": {
5        "close": {
6          "avg": 277430236.3636364,
7          "highest": 277800000.0,
8          "lowest": 277170000.0
9        },
10       "SMA_5": {
11         "avg": 277150965.8181818,
12         "highest": 277502506.0,
13         "lowest": 276346944.0
14       },
15       "EMA_10": {
16         "avg": 277018267.9226054,
17         "highest": 277289505.1079296,
18         "lowest": 276704324.598024
19       },
20       "RSI_10": {
21         "avg": 65.07231402887675,
22         "highest": 72.10420259192908,
23         "lowest": 50.9614355559214
24       },
25       "gain_loss": {
26         "highest_gain": 1734569.0,
27         "highest_loss": 569950.0
28     }
```

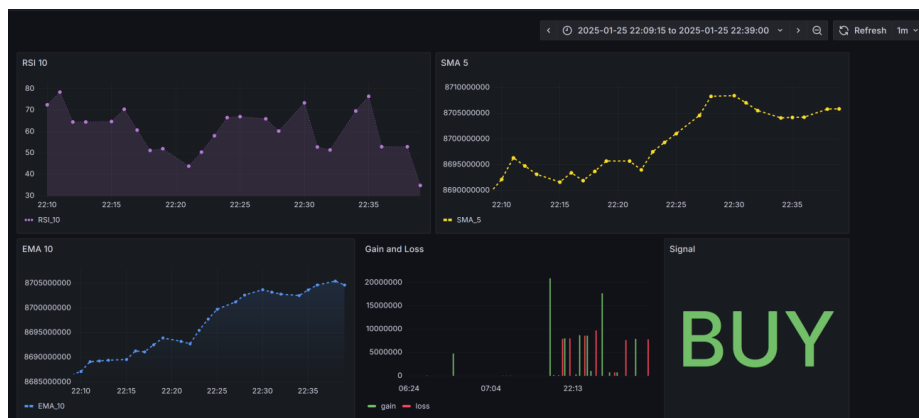
Summarize Multiple API

لیستی از چند نماد سهام را به عنوان ورودی گرفته و خلاصه عملکرد را برای هر کدام در بازه زمانی مشخص تولید می‌کند.

Visualization Service (Grafana)

در سرویس گرافانا که برای نمایش نمودارها و چارت‌ها استفاده شده است این قابلیت وجود دارد که به صورت socket-based برای دریافت سیگنال و همچنین query-based برای انجام عملیات aggregation استفاده شود.

در این سرویس در گرافانا یک داشبورد برای هر سهام ساخته شده است که در یک نگاه خلاصه‌ای از عملکرد سهام و نمودارهای آن نشان داده شده است که به شرح زیر است:



قابلیت نشان دادن سیگنال به صورت کوثری و یا استفاده از WebSocket که به شرح زیر است:

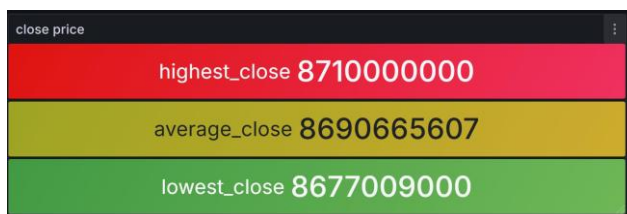


در نمودار زیر با استفاده از مقادیر دریافتی سهام، Candlestick آن را در گرافانا ساخته و مقادیر SMA، EMA را نمایش می‌دهیم:

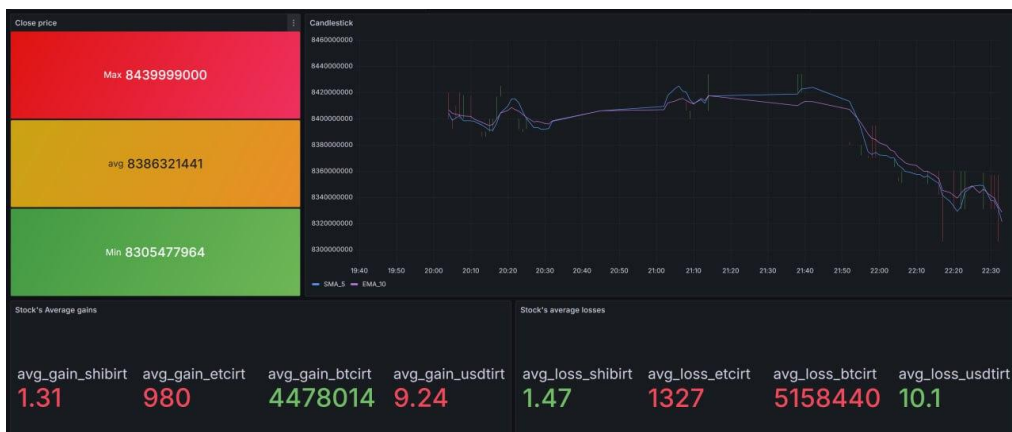


«خط بنفش نشان‌دهنده EMA و خط آبی نشان‌دهنده SMA است»

همچنین اعمال عملیات Aggregation برای نمایش خلاصه‌ای از عملکرد سهام:



همچنین در داشبورد زیر میانگین سود تو سهام‌های مختلف و میانگین ضرر در سهام‌های مختلف آورده شده است:



Trending Prediction Service

در این سرویس اسکریپت پایتونی قرار گرفته است که به طور کلی سیستمی برای پیشبینی قیمت سهام به کمک مدل شبکه عصبی بازگشتی LSTM پیاده‌سازی می‌کند. در بخش‌های بعدی این الگوریتم بیشتر توضیح داده خواهد شد. در ادامه بخش‌های مختلف کد به صورت مفصل‌تری توضیح داده خواهد شد:

وارد کردن کتابخانه‌های مورد نیاز

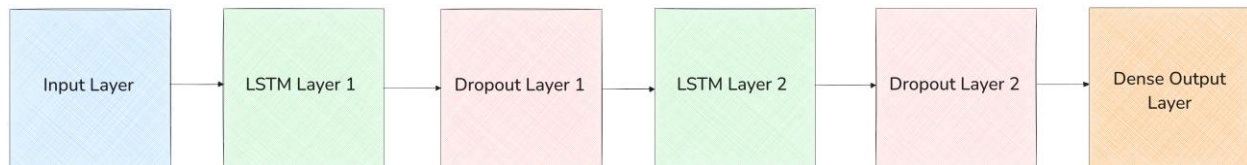
- کتابخانه NumPy و Pandas برای کار با داده‌ها و عملیات عددی استفاده می‌شود.
- کتابخانه Matplotlib برای ترسیم نمودارها
- کتابخانه Keras برای ساخت و آموزش مدل LSTM
- کتابخانه yfinance برای دانلود داده‌های به روز بازار سهام
- کتابخانه argparse و os برای دریافت نماد سهام از طریق خط فرمان یا متغیرهای محیطی
- کتابخانه time و datetime برای کنترل زمان اجرای کد

تابع ساخت مدل LSTM

در این بخش با تابعی به اسم `build_lstm_model` برای آموزش ساخته می‌شود.

- مدلی با دو لایه LSTM و لایه‌های Dropout برای جلوگیری از بیش برازش (overfitting) ایجاد می‌کند.
- یک لایه Dense خروجی برای پیشبینی مقدار نهایی قرار گرفته است.

توضیحات معماری مدل به ترتیب لایه‌ها به شرح زیر است:



- لایه ورودی: داده‌های ورودی شامل ۶۰ گام زمانی و ۱ ویژگی است. (قیمت close در ۶۰ روز گذشته)
- لایه LSTM اول: شامل ۵۰ واحد LSTM با `return_sequences = True` که خروجی به لایه بعدی ارسال می‌گردد.
- لایه Dropout اول: با نرخ ریزش ۲۰ درصد برای جلوگیری از بیش برازش
- لایه LSTM دوم: شامل ۵۰ واحد LSTM با `return_sequences = False` که تنها آخرین حالت را به خروجی می‌دهد.
- لایه Dropout دوم: با نرخ ریزش ۲۰ درصد برای جلوگیری از بیش برازش
- لایه Dense خروجی: یک واحد خروجی که قیمت بسته شدن سهام را پیشبینی می‌کند.

تابع دانلود داده‌های سهام

این تابع که `fetch_stock_data` نام دارد با استفاده از کتابخانه `yfinance` داده‌های قیمت باز و بسته شدن، کمینه و بیشینه سهام را برای بازه زمانی مورد نظر دریافت می‌کند.

تابع آماده سازی داده‌ها (prepare_data)

- داده‌ها را نرمال‌سازی می‌کند تا مدل بهتر آموزش ببیند.
- دنباله‌های زمانی به طول ۶۰ روز را به عنوان ورودی (x_train) و قیمت بسته شدن را به عنوان خروجی (y_train) آماده می‌کند.

تابع بازگرداندن داده‌ها به مقیاس اصلی (inverse_transform)

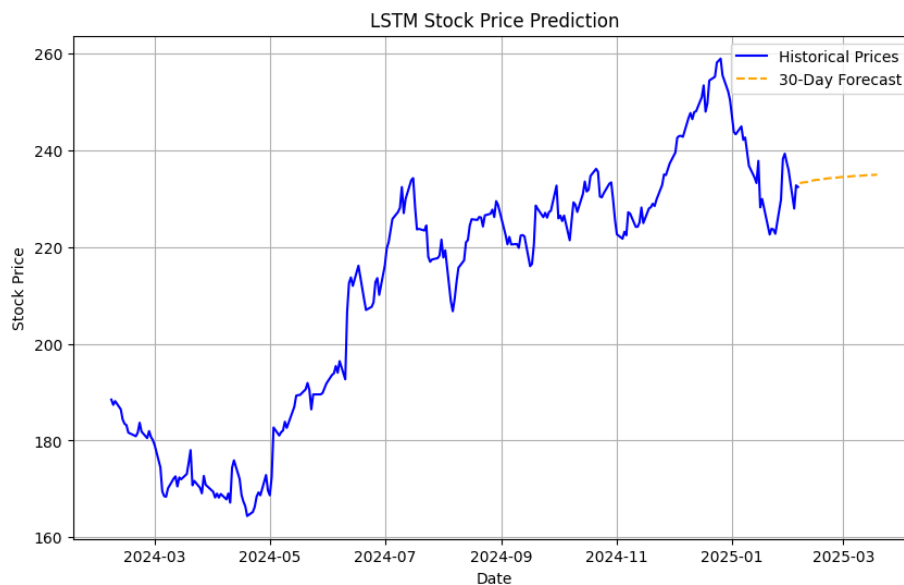
پیشبینی‌های نرمال‌سازی شده را به مقیاس واقعی قیمت باز می‌گرداند.

تابع پیشبینی آینده (predict_future)

- به کمک داده‌های اخیر و مدل آموزش‌دیده، پیشبینی قیمت ۳۰ روز آینده را انجام می‌دهد.
- از داده‌های پیشبینی شده به عنوان ورودی برای پیشبینی‌های بعدی استفاده می‌کند.

تابع رسم نمودار (plot_dynamic_symbol)

قیمت‌های ۶۰ روز گذشته و پیشبینی‌های ۳۰ روز آینده را در یک نمودار ترسیم می‌کند مانند شکل زیر:



حلقه اصلی اجرای کد (__main__)

- هر روز داده‌های سهام را به روز کرده و مدل را دوباره آموزش می‌دهد.
 - داده‌های پیشبینی شده را روی نمودار نمایش می‌دهد.
 - کد به طور خودکار هر ۲۴ ساعت (این مقدار قابل تنظیم است) اجرا شده و نتایج نمودار به روزرسانی می‌گردد.
- بنابراین این اسکریپت پایتونی به صورت خودکار داده‌های قیمت سهام را در ۶۰ روز گذشته به دست آورده، مدل LSTM را آموزش داده و سپس پیشبینی ۳۰ روز آینده را به همراه نمودار به روز می‌کند.

تکنولوژی‌های استفاده شده

API Nobitex

یک رابط برنامه نویسی کاربردی است که به شما اجازه می‌دهد تا اطلاعات بازار رمز ارز را از صرافی Nobitex دریافت نمایید. این API می‌تواند داده‌هایی همچون قیمت‌ها، حجم معاملات، نرخ تغییرات و اطلاعات دیگر از بازار رمز ارزها را در اختیار شما قرار دهد.

نحوه دریافت داده

از طریق API Nobitex می‌توان درخواست‌هایی ارسال کرد که داده‌های مختلف مانند قیمت رمز ارزها و حجم معاملات، بالاترین و پایین‌ترین قیمت‌ها و تغییرات قیمت در یک بازه زمانی مشخص را دریافت کند. این داده‌ها معمولاً در قالب JSON ارسال می‌شوند که به راحتی می‌توان آن‌ها را پردازش و در سیستم‌های دیگر مورد استفاده قرار داد.

چرا این API انتخاب شده است؟

- **دقت بالا:** Nobitex یک صرافی معتبر در ایران است که داده‌های دقیق و به روز بازارهای رمز ارز را ارائه می‌دهد.
- **سرعت بالا:** این API دارای سرعت بسیار بالایی در ارسال و دریافت داده دارد که برای تحلیل بلادرنگ بسیار حیاتی می‌باشد.
- **قابلیت‌های خاص:** این API امکان دسترسی به انواع داده‌ها را فراهم می‌آورد که برای تجزیه و تحلیل مناسب است.

Apache Kafka

یک سیستم پیام‌رسان توزیع شده و مقیاس‌پذیر است که برای ارسال، دریافت و ذخیره‌سازی داده استفاده می‌شود. Kafka به ویژه برای پردازش داده‌ها به صورت Realtime و در مقیاس بزرگ به کار می‌رود.

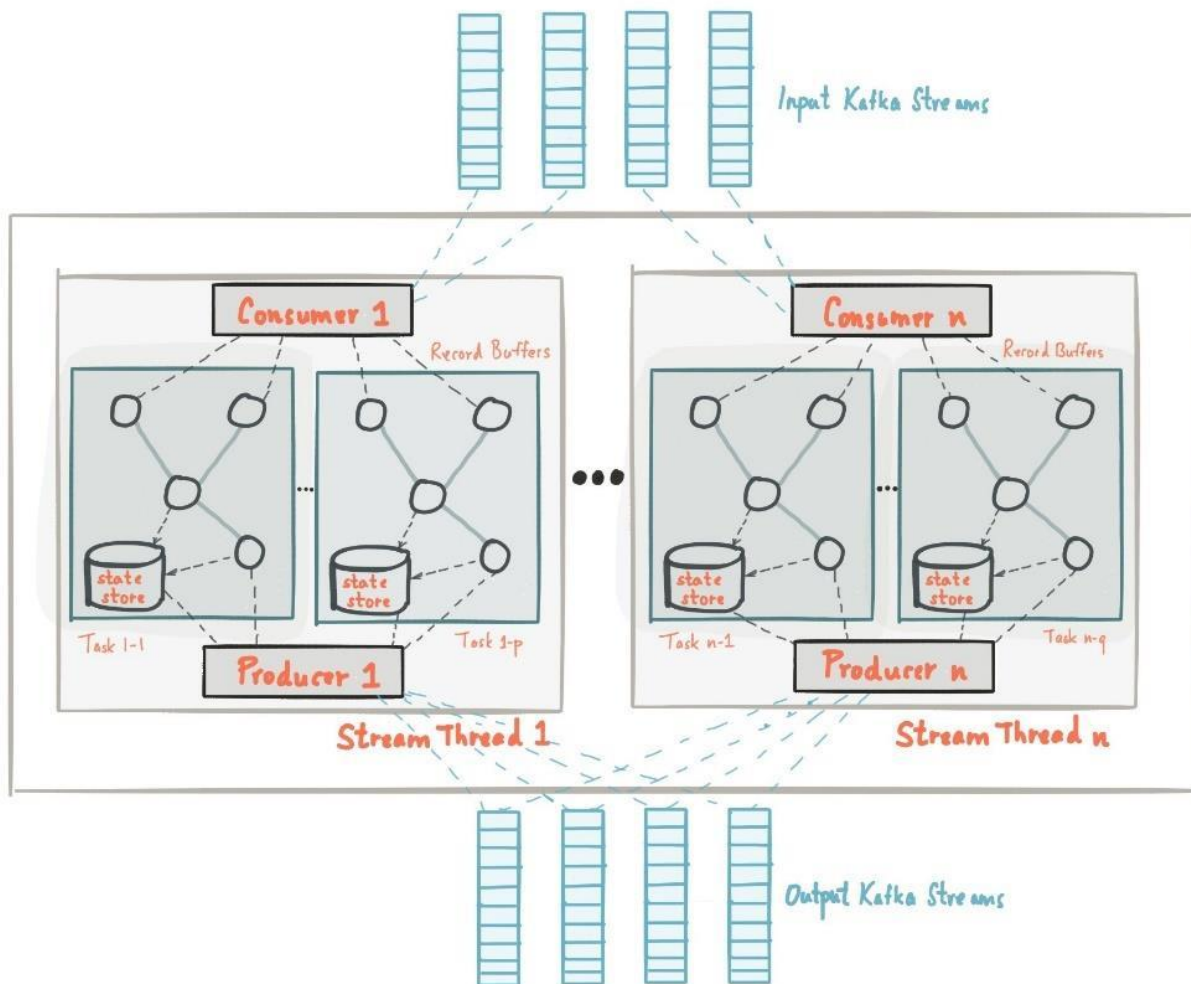
چرا Kafka انتخاب شده است؟

- **مقاوم در برابر خرابی‌ها (Fault Tolerance):** Kafka قادر است داده‌ها را در صورت بروز مشکلات در سیستم‌های مختلف نگهداری کرده و از دست رفتن داده‌ها جلوگیری کند. این ویژگی در پروژه‌هایی که نیاز به پردازش داده‌های حیاتی دارند، ضروری است.
- **مقیاس‌پذیری:** Kafka قادر است حجم عظیمی از داده‌ها را به طور همزمان پردازش کرده و بدون کاهش کارایی، داده‌ها را به نودهای مختلف ارسال کند. این ویژگی برای سیستم‌های بلادرنگ و توزیع شده مانند پروژه‌های مالی بسیار مناسب است.

معماری Kafka

- **Producer:** این نودها مسئول ارسال داده‌ها به Kafka هستند. در این پروژه، میکروسرویس‌ها یا نودهای پردازش داده‌ها می‌توانند به عنوان Producer عمل کنند.
- **Broker:** داده‌ها را دریافت کرده و آن‌ها را در Topic‌ها ذخیره می‌کنند. این داده‌ها به نودهای مختلف (Consumer) ارسال می‌شوند.
- **Consumer:** این نودها مسئول دریافت و پردازش داده‌ها از Topic‌ها هستند. در این پروژه، Spark و سایر سیستم به عنوان Consumer داده‌ها را پردازش کنند.

در شکل زیر معماری Kafka به صورت شماتیک آورده شده است:



Apache Spark

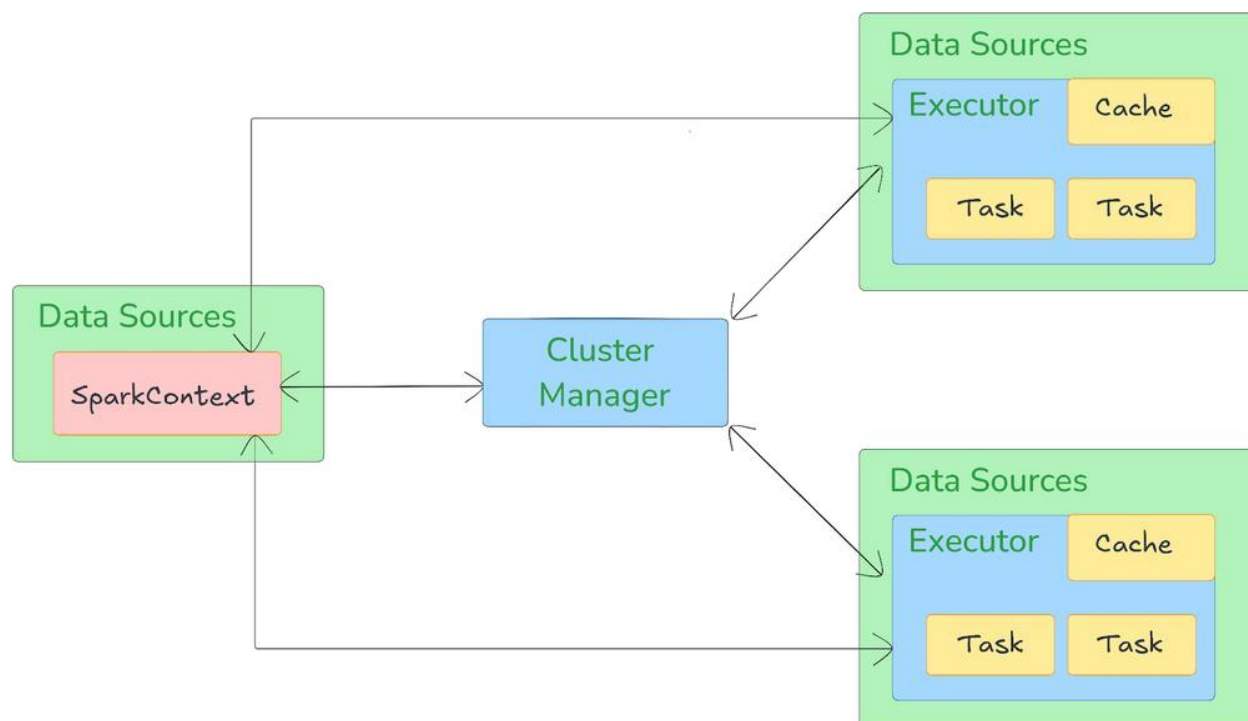
یک فریم‌ورک پردازش داده‌های توزیع شده است که قادر است داده‌ها را به صورت موازی و در مقیاس بزرگ پردازش کند. Spark در این پروژه برای تحلیل بلادرنگ داده‌های مالی به کار می‌رود.

چرا اسپارک انتخاب شده است؟

- **پردازش بلادرنگ:** Spark به راحتی از داده‌های استریم پشتیبانی می‌کند و قادر است آن‌ها را در لحظه پردازش کند. این ویژگی به ویژه برای تحلیل داده‌های مالی که نیاز به پردازش سریع دارند، بسیار مفید است.
- **مقیاس‌پذیری:** Spark می‌تواند داده‌های حجیم را به صورت توزیع شده پردازش کند و به راحتی با افزایش حجم داده‌ها مقیاس‌پذیر می‌شود.
- **پردازش داده‌های استریم:** Spark Streaming امکان پردازش داده‌های ورودی به صورت پیوسته را فراهم می‌کند. این ویژگی برای سیستم‌هایی که نیاز به تحلیل بلادرنگ دارند، ضروری است.
- **دریاچه داده‌های توزیع شده:** Spark می‌تواند داده‌ها را در مقیاس وسیع توزیع کرده و پردازش کند، که در پروژه‌هایی با حجم بالای داده‌ها بسیار ضروری است.

معماری Spark

- **Driver:** کنترل کننده اصلی سیستم که نودها و منابع را مدیریت می‌کند. Driver تصمیم می‌گیرد که چه داده‌هایی باید پردازش شوند و به چه نودی ارسال شوند.
- **Executor:** این نودها مسئول انجام پردازش‌ها و محاسبات واقعی هستند.
- **Cluster Manager:** این بخش برای مدیریت منابع (مثل حافظه و پردازنده‌ها) در کلاستر Spark استفاده می‌شود.



QuestDB

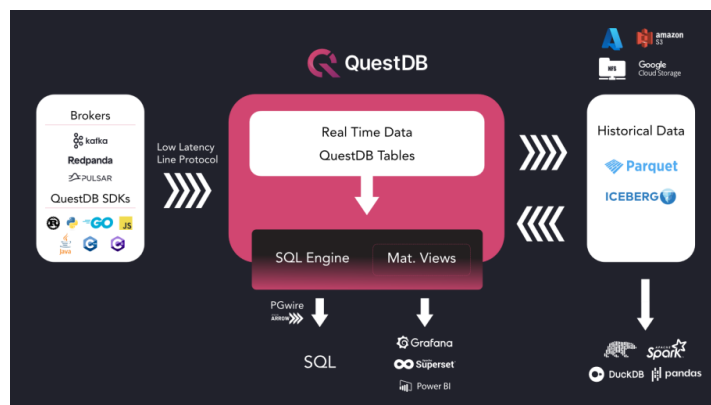
QuestDB یک دیتابیس بهینه‌شده برای داده‌های زمانی است. این پایگاه داده به ویژه برای پردازش و ذخیره‌سازی داده‌هایی که به ترتیب زمانی ثبت می‌شوند، مانند داده‌های مالی و بازارهای رمز ارز، طراحی شده است.

چرا از QuestDB استفاده شده است؟

استفاده از این دیتابیس بسیار بهینه‌تر از ترکیب یک دیتابیس دیگر مثل PostgreSQL با Redis است زیرا:

- **عملکرد بالا:** QuestDB برای پردازش داده‌های سری زمانی بهینه شده و می‌تواند حجم عظیمی از داده‌ها را با سرعت بسیار بالا وارد کوئری کند.
- **کاهش پیچیدگی معماری:** با استفاده از QuestDB دیگر نیازی به ترکیب چندین ابزار نیست.
- **مقیاس‌پذیری:** QuestDB در مدیریت حجم عظیمی از داده‌های سری زمانی مقیاس‌پذیری بالایی دارد، در حالی که Redis برای داده‌های بسیار بزرگ نیاز به حافظه بیشتر و تنظیمات پیچیده‌تری دارد.
- **پشتیبانی از ILP:** امکان ارسال داده‌ها با فرمت ILP باعث ساده‌تر شدن ارسال داده‌های بازار سهام می‌شود.

- **جستجو و انجام کوئری‌های پیشرفته SQL:** قابلیت استفاده از SQL استاندارد برای کوئری‌های پیچیده مانند محاسبه میانگین، محاسبه شاخص‌های معاملاتی (...، SMA, EMA) و جستجوی مقادیر خاص در بازه‌های زمانی
- **پشتیبانی از داده‌های بلادرنگ (Realtime):** مناسب برای سیستم‌های بلادرنگ که داده‌های بازار سهام را در لحظه پردازش و تحلیل می‌کنند.
- **مقیاس‌پذیری بالا:** مناسب برای ذخیره و پردازش داده‌های حجیم مانند داده‌های تاریخی بازار سهام
- **پشتیبانی از چارت‌های مختلف برای تحلیل بصری داده**

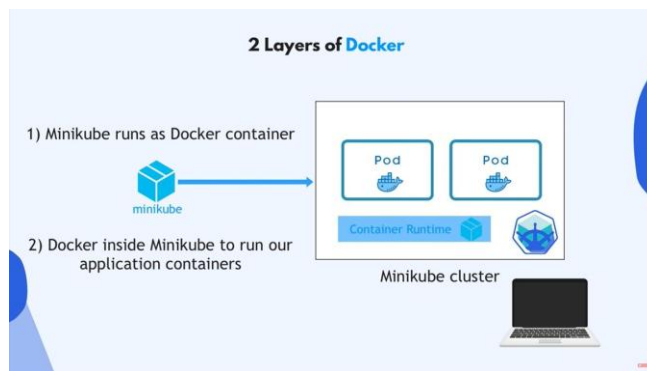


Minikube

ابزاری برای شبیه‌سازی کلاسترهای Kubernetes به صورت محلی طراحی شده است. این ابزار به شما این امکان را می‌دهد که کلاسترهای توزیع شده را در محیط‌های محلی تست و توسعه دهید.

چرا Minikube انتخاب شده است؟

- **شبیه‌سازی محیط‌های Kubernetes محلی:** Minikube به توسعه‌دهندگان این امکان را می‌دهد که بدون نیاز به محیط‌های ابری، کلاسترهای Kubernetes را در محیط محلی خود شبیه‌سازی کنند.
- **مدیریت کلاسترهای توزیع شده:** Minikube امکان مدیریت و تست آسان کلاسترهای توزیع شده را فراهم می‌آورد که برای تست و توسعه سیستم‌های بزرگ و پیچیده مانند این پروژه ضروری است.
- **صرفه‌جویی در هزینه‌ها:** استفاده از Minikube به جای ایجاد کلاسترهای Kubernetes در محیط‌های ابری می‌تواند به طور قابل توجهی هزینه‌های زیر ساختی را کاهش دهد.



الگوریتم LSTM

الگوریتم Long Short Term Memory یا به اختصار LSTM یکی از انواع شبکه‌های عصبی بازگشتی (RNN) است که برای پردازش داده‌های دنباله‌دار و سری‌های زمانی مناسب است. مشکل اصلی شبکه‌های RNN کلاسیک، فراموش کردن اطلاعات طولانی مدت و مشکل نابودی گرادیان (Vanishing Gradient) در یادگیری است. LSTM این مشکل را با معرفی حافظه طولانی مدت و مکانیزم دروازه‌ای (gating mechanism) حل می‌کند.

ساختار LSTM

LSTM از سه دروازه (Gate) استفاده می‌کند که برای کنترل جریان اطلاعات ورودی و خروجی طراحی شده است:

1. **دروازه فراموشی (Forget Gate):** تصمیم می‌گیرد که چه مقدار از اطلاعات قبلی حذف شود.
2. **دروازه ورودی (Input Gate):** تصمیم می‌گیرد که چه مقدار از اطلاعات جدید وارد حافظه شود.
3. **دروازه خروجی (Output Gate):** تعیین می‌کند که چه مقدار از حافظه به خروجی فرستاده شود.

این مکانیزم باعث می‌شود که LSTM بتواند وابستگی طولانی مدت را حفظ کند و اطلاعات مهمی را در طول زمان به خاطر بسپارد.

چرا LSTM برای تحلیل داده‌های مالی مناسب است؟

- **مدیریت داده‌های سری زمانی:** داده‌های مالی مانند قیمت سهام، نرخ ارز و حجم معاملات دارای الگوهای زمانی پیچیده هستند که LSTM می‌تواند وابستگی‌های طولانی مدت را یاد گرفته و روندهای گذشته را برای پیش‌بینی آینده استفاده کند.
- **حل مشکل نابودی گرادیان:** در داده‌های مالی، اطلاعات قدیمی همچنان می‌توانند روی روند آینده تاثیر بگذارند. LSTM برخلاف RNN معمولی، قادر به یادگیری روابط طولانی مدت بین داده‌های گذشته و آینده است.
- **پیش‌بینی روندها و الگوهای تکرار شونده:** بازارهای مالی معمولاً دارای الگوهای چرخه‌ای و روندی هستند. LSTM قادر است این الگوها را شناسایی کند و به پیش‌بینی نوسانات بازار کمک کند.
- **انعطاف‌پذیری در پردازش ویژگی‌های مختلف:** LSTM می‌تواند چندین ویژگی مانند حجم معاملات، اخبار اقتصادی و احساسات بازار و سایر داده‌های مرتبط را پردازش کند که ما در این سیستم صرفاً از داده‌های خام مالی استفاده کرده‌ایم.
- **کاربرد در الگوتریدینگ (Algo-Trading):** بسیاری از الگوریتم‌های معاملاتی خودکار از LSTM برای تحلیل داده‌های بازارهای مالی و تصمیم‌گیری لحظه‌ای استفاده می‌کنند.

چالش‌های استفاده از LSTM در تحلیل مالی

1. **حساسیت به نویز:** داده‌های مالی شامل نویز و نوسانات زیادی هستند که می‌تواند بر دقت مدل تاثیر بگذارد.
2. **نیاز به داده‌های زیاد:** برای آموزش یک مدل LSTM کارآمد، حجم زیادی از داده‌های تاریخی نیاز است.
3. **محاسبات سنگین:** این مدل نسبت به مدل‌های ساده شبکه عصبی نیاز به پردازش سنگین‌تری دارد.

راهکارهای افزایش دقت مدل

برای افزایش دقت مدل راه‌های زیادی وجود دارد اما ساده‌ترین و موثرترین آنها به شرح زیر است:

- **افزایش داده‌های آموزش:** بهتر است به جای ۶۰ روز داده‌های ۱۲۰ روز یا بیشتر برای آموزش مدل داده شود.
- **تعیین دقیق epoch‌های مدل:** بهتر است که برای تعیین تعداد epoch مناسب چندین بار آزمون و خطا صورت گیرد.
- **افزایش لایه‌های LSTM:** برای افزایش دقت مدل و شناسایی بهتر روند می‌تواند تعداد لایه‌های LSTM را افزایش داد.

الگوریتم‌های معاملاتی (اندیکاتورها)

در این سیستم از سه الگوریتم معاملاتی پرکاربرد استفاده شده است که در ادامه به بررسی هر کدام خواهیم پرداخت.

میانگین متحرک ساده (SMA)

SMA یکی از پرکاربردترین روش‌ها برای تعیین روند قیمت است. این شاخص میانگین قیمت یک سهام را در یک بازه زمانی مشخص (پنجره زمانی) محاسبه می‌کند. برای این کار، قیمت‌های پایانی جمع شده و سپس بر تعداد پنجره زمانی تقسیم می‌گردد.

فرمول محاسبه SMA به شرح زیر است:

$$\frac{P_1 + P_2 + \dots + P_n}{n} = SMA$$

- در این فرمول P_t قیمت پایانی (close) در پنجره مورد نظر است. (در این سیستم بازه زمانی به صورت دقیقه است)
- در این فرمول n تعداد پنجره‌های انتخابی است که در این سیستم ۵ در نظر گرفته شده است. (پنجره زمانی ۵ دقیقه است)

میانگین متحرک نمایی (EMA)

EMA مشابه SMA عمل می‌کند اما به داده‌های اخیر وزن بیشتری می‌دهد. این موضوع باعث می‌شود EMA نسبت به تغییرات قیمتی سریعتر واکنش دهد. این الگوریتم در معاملات کوتاه مدل و برای شناسایی تغییرات سریع کاربرد دارد.

فرمول محاسبه EMA به شرح زیر است:

$$EMA_t = P_t \times \alpha + EMA_{t-1} \times (\alpha - 1)$$

- در این فرمول α ثابت هموارسازی است.
- در این فرمول P_t قیمت پایانی فعلی است.
- در این فرمول n تعداد پنجره‌های انتخابی است که در این سیستم ۱۰ در نظر گرفته شده است. (پنجره زمانی ۱۰ دقیقه است)

شاخص قدرت نسبی (RSI)

RSI یک شاخص مومنتوم است که سرعت و تغییرات حرکات قیمتی را اندازه می‌گیرد. این شاخص معمولاً بین ۰ تا ۱۰۰ نوسان می‌کند و به معامله‌گران کمک می‌کند شرایط اشباع خرید یا اشباع فروش را تشخیص دهند. مقادیر بالای ۷۰ نشان‌دهنده اشباع خرید و مقادیر زیر ۳۰ نشان‌دهنده اشباع فروش است.

فرمول محاسبه RSI به شرح زیر است:

$$RS = \frac{Avg \cdot Gain}{Avg \cdot Loss}$$

$$RSI = 100 - \frac{100}{1 + RS}$$

- میانگین سودهای مثبت و زیان‌های منفی در این سیستم در بازه زمانی (پنجره زمانی) ۱۰ روزه محاسبه می‌شود.
- همانطور که پیشتر هم توضیح داده شد این سه شاخص با هم ترکیب شده و سیگنال‌های خرید، فروش و نگهداشتن تولید شده و به معامله‌گران در شناسایی نقاط ورود و خروج مناسب در بازار کمک می‌کند.

نتیجه گیری

سیستم طراحی شده با استفاده از Django، QuestDB، Spark، Kafka و اسکریپت پایتونی با الگوریتم LSTM قادر به پردازش سریع و دقیق داده‌های به صورت Realtime و در مقیاس بزرگ است که می‌تواند علاوه بر تولید سیگنال‌های معاملاتی به صورت لحظه‌ای به طور موثر عمل کند، بلکه می‌تواند با استفاده از الگوریتم LSTM قیمت پایانی هر سهامی را تا ۳۰ روز آینده پیش‌بینی کند.

با این حال ممکن است این پروژه چالش‌هایی همچون تاخیر، مقیاس‌پذیری و عدم دقت کافی و مناسب در نتایج الگوریتم LSTM داشته باشد که برای این موارد، با بهینه‌سازی‌های مختلفی می‌توان عملکرد سیستم را بهبود بخشید.

کدهای این پروژه بر روی گیت‌هاب زیر قرار دارد.

The screenshot shows the GitHub repository page for 'Big-Data-Engineering-Financial-Analysis' by AliAhmadi-Software. The repository is public and has 2 branches and 0 tags. The commit history shows a series of updates, including adding documents, modifying project files, and adding initial configurations. The repository also includes a README file and a list of contributors.

Commit	Message	Time
ceff348	Fixed Error + Add Document	19 hours ago
	Modify Project-Docment	9 hours ago
	qdb init added	2 days ago
	add yaml files	3 days ago
	added avg gain, avg loss, avg close panels to grafana	13 hours ago
	add stream processing service	2 days ago
	cmd.sh	3 days ago
	initial configs added	2 days ago
	initial configs added	2 days ago
	fixed dashboards importing problem	14 hours ago
	initial configs added	2 days ago

Contributors: AliAhmadi-Software, Ahooraa Ahoora Amini, Mahdishk

Languages: Jupyter Notebook 55.7%, Python 39.0%

با تشکر از استاد نادری و استاد انتظاری که ما را در پیشبرد این پروژه یاری کردند.