

第四周 时间序列大数据分析作业1

课堂上，我们试了试 比特币，以太币大数据的分析。

1. Excel 有局限，打不开
2. GPT等AI也有局限，无法运行
3. 所以，往往高级分析需要Python。你虽然不太会，但有AI帮助你。

作业：

大家可以使用 比特币，以太币这个大数据，指定一个问题，比如用那段时间，来预测哪一天的价格等。（当然与课堂一起练习的时间不能一致）

然后，找到用什么方法，来预测最准确。你为何说你找到的方法是最好的。（也许因为你比较了多个方法）

算法可以使用LSTM。当然，你想换更好更准确的算法，用其他大数据，也允许。

最重要的1：看你有没有认真思考，有没有认真努力尝试。

最重要的2：你有没有努力，将你所理解的，尝试给别人讲清楚。

提交：**你做的代码文件+用PPT整理你做的过程与思路。**

第4周 上课时，彼此分享心得体会+发表

个人作业 40分

数据量很大让python帮我们提取想要的数据

ID	Date	Unix timestamp	Open	High	Low	Close	Volume
1048566	2019/8/14 20:49	1565815740	188.07	188.07	188.07	188.07	2.88166405
1048567	2019/8/14 20:50	1565815800	188.15	188.15	188.15	188.15	0.26537337
1048568	2019/8/14 20:51	1565815860	188.15	188.15	188.15	188.15	2.935
1048569	2019/8/14 20:52	1565815920	188.51	188.71	188.47	188.71	12.7531055
1048570	2019/8/14 20:53	1565815980	188.97	189.1	188.83	189.1	66.663086
1048571	2019/8/14 20:54	1565816040	189.23	189.26	189.23	189.26	3.06512211
1048572	2019/8/14 20:55	1565816100	189	189	188.05	188.05	201.894447
1048573	2019/8/14 20:56	1565816160	188.03	188.03	188.03	188.03	7.0262419
1048574	2019/8/14 20:57	1565816220	187.58	187.58	187.44	187.44	54.1548724
1048575	2019/8/14 20:58	1565816280	187.68	188.17	187.68	188.17	6.63301562
1048576	2019/8/14 20:59	1565816340	188.28	188.28	188.28	188.28	1.81

Microsoft

帮我写一个代码。提取文件以太币有时间.xlsx。时间从2019年6月10日零点开始到7月10日零点为止的所有数据。信息包括 datetime, open, close, low 和 volume。且自动帮我生成excel 表格。谢谢

既然你的 Java 环境在读取 .xlsx 文件时遇到问题，通常是因为该软件对 Excel 压缩格式的支持不佳。针对 2019 年 6 月 10 日到 7 月 10 日的数据提取，我为你编写了一个更稳健的代码，它会生成一个 .csv 文件。

CSV 格式是纯文本，兼容性最高，基本能解决所有 Java 环境下的 No loader 报错。同时，我修正了时间范围和可能存在的列名拼写问题。

数据提取代码 (导出为兼容性最高的 CSV)



将 CSV 文件转换为 Excel 文件 (.xlsx) 在 Python 中非常简单，主要使用 `pandas` 库。

1. 最简单的转换代码

如果你已经安装了 `pandas` 和 `openpyxl`，只需几行代码即可完成转换：

Python



```
import pandas as pd

# 1. 读取 CSV 文件
# encoding='utf-8-sig' 可以防止中文乱码
df = pd.read_csv('你的文件.csv', encoding='utf-8-sig')

# 2. 导出为 Excel 文件
# index=False 表示不保存行索引 (即左侧的 0, 1, 2...)
df.to_excel('转换后的文件.xlsx', index=False)
```



问AI有哪些模型

我想用比特币的一段时间的数据去预测未来的价格，可以用哪些方法

用一段时间的比特币历史数据来预测未来价格。常见的方法可以分为 传统统计方法、机器学习方法、深度学习方法、以及混合/强化学习方法。下面我按「从简单到复杂、从可解释到高预测力」给你一个清晰的体系，并结合加密货币的特点说明适用场景。

支持向量回归 (SVR)

- 对小样本、非线性有效
- 对参数敏感

树模型（非常适合加密货币）

强烈推荐

- Random Forest
- XGBoost
- LightGBM
- CatBoost

优点

- 捕捉非线性
- 不需要平稳化
- 抗噪声

特征示例

```
text
[BTC(t-1), BTC(t-5), BTC(t-60), Volume, RSI, MACD, ATR]
```

给我介绍下什么是random forest

Random Forest（随机森林）是一种集成学习（Ensemble Learning）方法，属于基于决策树的机器学习算法，在金融、时间序列、分类和回归问题中都非常常用，尤其适合你这种比特币价格预测场景。

下面我从「直觉 → 原理 → 数学思想 → 优缺点 → 金融应用」系统给你讲清楚。

• Random Forest

一、一句话直觉理解

随机森林 = 很多“有点不一样”的决策树投票（或取平均）

- 每棵树都不完美
- 但很多不完美的树加在一起，反而很强



二、它解决了什么问题？

决策树的两大缺点

1. 容易过拟合
2. 对数据扰动敏感（不稳定）

👉 Random Forest 用“随机性 + 集成”解决了这两个问题。



三、Random Forest 的核心思想（非常重要）

随机森林的“随机”主要体现在 两个地方：

① 样本随机 (Bootstrap)

- 从原始训练集 **有放回抽样**
- 每棵树看到的数据都不一样

text

复制代码

原始数据：1000 条

每棵树：随机抽 1000 条（可重复）

② 特征随机 (Feature Subsampling)

- 每次节点分裂时
- 只从随机选取的一部分特征中找最优分裂

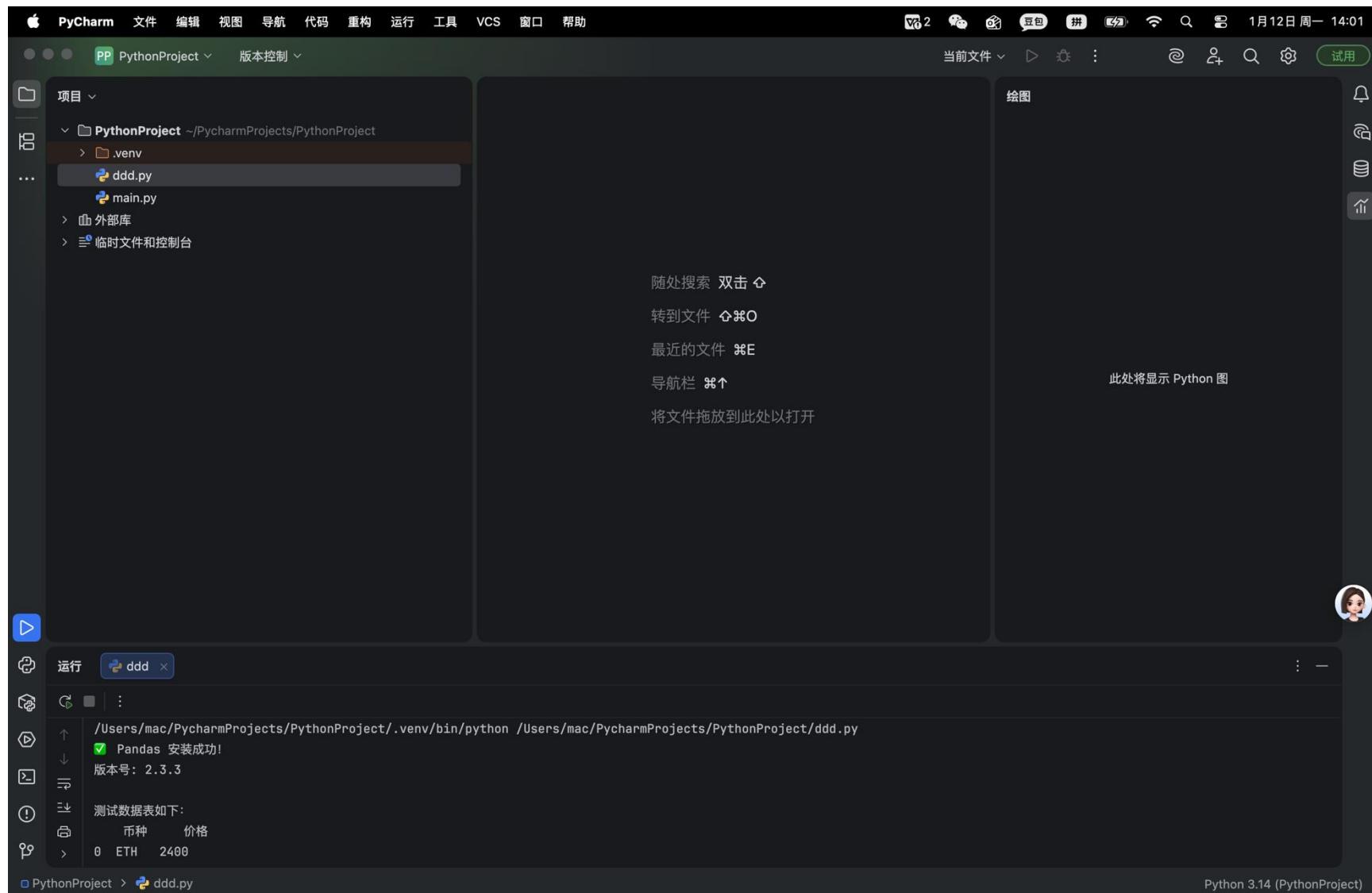
text

复制代码

总特征数 = 20

PyCharm

- Python相当于语言比如说
是中文/英文， PyCharm是IDE
的一种， 相当于不同的办公
软件有微软， 苹果， 谷歌和
WPS。



Python跑不同的模型实际需要很多的库，AI帮你编代码的时候会写入，运行后报错，把结果给AI。AI会告诉你缺少哪些库，会给你代码。PyCharm也会提示你，也可以直接在里面点安装程序（基本所有的库都有，有些Python的版本过高，有些库没有适配的版本，如果这个库是会影响模型的使用就需要切换Python的版本）。

怎么把excel放入python分析

要把 Excel 文件放入 Python 进行分析，最常用的工具就是你刚刚安装好的 **Pandas**。

由于你之前提到过文件名为 `第四周大数据分析作业.xlsx`，我为你写一段可以直接运行的代码。

1. 准备工作：安装必要库

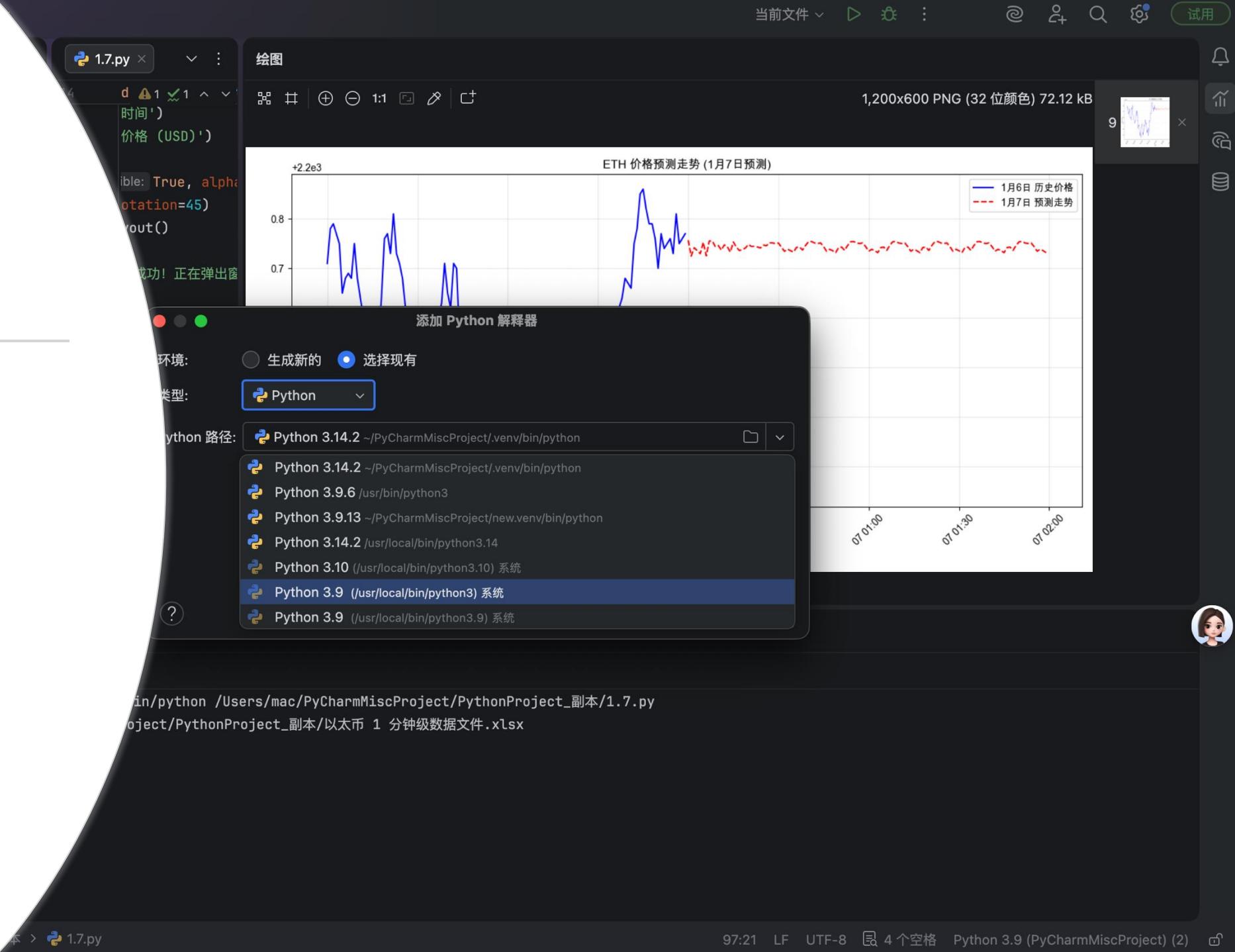
Pandas 读取 Excel 需要一个专门的引擎（`openpyxl`）。请在 PyCharm 的 **Terminal** 中输入：

Bash

```
pip install openpyxl -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2. Python 读取 Excel 的代码

将以下代码复制到你的 `main.py` 中。请确保你的 Excel 文件和 `.py` 文件放在同一个文件夹里。



- 更换Python的版本

安装库的程序



```
终端 本地 × 本地 (2) × + ▾
Downloading https://pypi.tuna.tsinghua.edu.cn/packages/c0/da/977ded879c29cbd04de313843e76868e6e13408a94ed6b987245dc7c8506/openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
)
Collecting et-xmlfile (from openpyxl)
Downloaded https://pypi.tuna.tsinghua.edu.cn/packages/c1/8b/5fe2cc11fee489817272089c4203e679c63b570a5aaeb18d852ae3cbba6a/et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-2.0.0 openpyxl-3.1.5
[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: pip install --upgrade pip
(.venv) mac@macdeMacBook-Air PythonProject %
```

1:1 LF UTF-8 4个空格 Python 3.14 (PythonProject)

有报错让AI继续改

LSTM 以太币价格预测分析

我的内容 >

Gem >

对话

LSTM 以太币价格预测分析

Python 分析 Excel 数据指南

Excel 快速提取大量数据技巧

特斯拉财报摘要与展望

以太币K线数据分析报告

发起新对话

Gemini

LSTM 以太币价格预测分析

Users/mac/PycharmProjects/PythonProject/.venv/bin/python /Users/mac/PycharmProjects/PythonProject/random forest.py
✓ 文件读取成功!

◆ 太棒了！你发出来的日志非常有价值，它直接告诉了我们问题出在哪里：

报错原因：你的 Excel 表格里，时间列的名字叫 `datetime`，价格列的名字叫 `close`（全部是小写）。但在代码里，你写的是 `x='Date'` 和 `y='Close'`（首字母大写了）。

Python 对大小写非常敏感，所以它找不到 `Date` 就报错了。

修复方案：直接修改绘图和数据提取行

请在你的 `random forest.py` 中找到对应行，按照下面的样子修改：

1. 修改绘图代码（针对你最后的报错）

将这一行：



项目

- PythonProject ~/PycharmProjects/PythonProject
 - .venv
 - bin
 - lib
 - share
 - .gitignore
 - CACHEDIR.TAG
 - pyvenv.cfg
- ddd.py
- main.py
- random forest.py
- 第四周大数据分析作业.xlsx

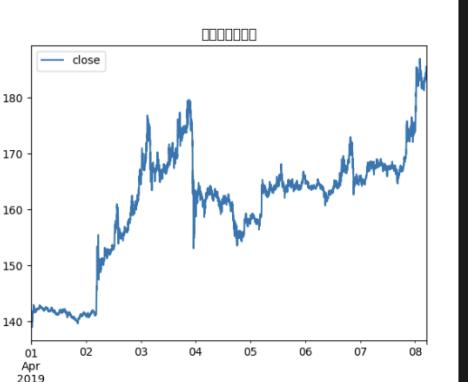
外部库
临时文件和控制台

想代码读取Excel文件，要把Excel文件和代码文件放在同一个文件夹里。

选择的数据4月1日到4月8日的以太币的走势

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 # 1. 设置文件路径
4 # 如果文件在同一个文件夹，直接写文件名；如果在别处，请写完整路径
5 file_path = '第四周大数据分析作业.xlsx'
6
7 try:
8     # 2. 读取 Excel 文件
9     # 如果有多个 Sheet，可以用 sheet_name='Sheet1' 指定
10    df = pd.read_excel(file_path)
11
12    # 3. 查看数据
13    print("✓ 文件读取成功!")
14    print("-" * 30)
15    print("数据前 5 行: ")
16    print(df.head()) # 显示前 5 行
17
18    print("-" * 30)
19    print("数据的基本信息: ")
20    print(df.info()) # 查看列名、数据类型、是否有缺失值
21
22 except FileNotFoundError:
23     print(f"✗ 错误：在当前目录下没找到文件 '{file_path}'")
24     print("请确认文件是否已经移动到 PyCharm 项目文件夹中。")
25 except Exception as e:
26     print(f"✗ 读取出错: {e}")
```

> 640x480 PNG (32位颜色) 26.15 kB



运行 random forest



:



/Applications/PyCharm.app/Contents/plugins/python-ce/helpers/pycharm_matplotlib_backend/backend_interagg.py:85: UserWarning: Glyph 36208 (\N{CJK UNIFIED IDEOGRAPH-8D70}) m:

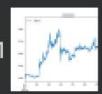
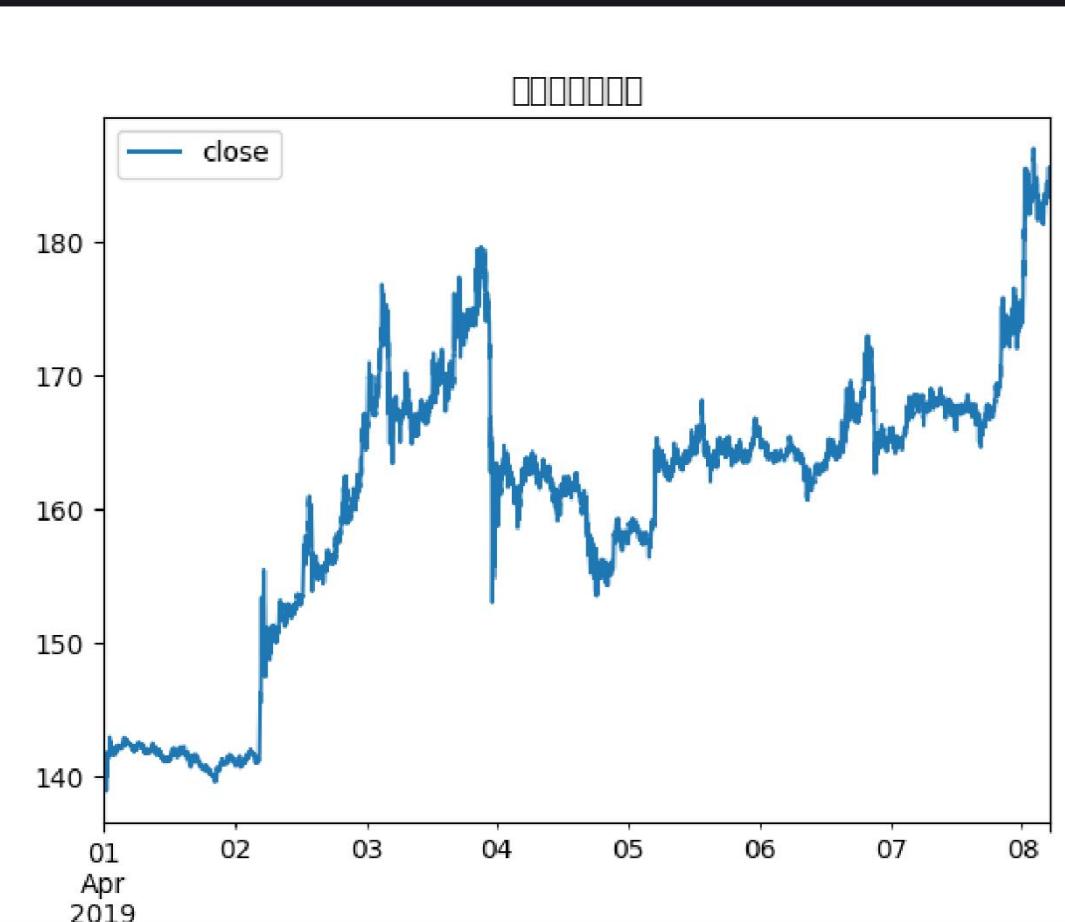
self.print_png(buf)

/Applications/PyCharm.app/Contents/plugins/python-ce/helpers/pycharm_matplotlib_backend/backend_interagg.py:85: UserWarning: Glyph 21183 (\N{CJK UNIFIED IDEOGRAPH-52BF}) m:

self.print_png(buf)

绘图

640x480 PNG (32 位颜色) 26.15 kB





项目

PythonProject ~/PycharmProjects/PythonProject

> .venv

ddd.py

main.py

random forest.py

random forest forcast.py

第四周大数据分析作业.xlsx

> 外部库

> 临时文件和控制台

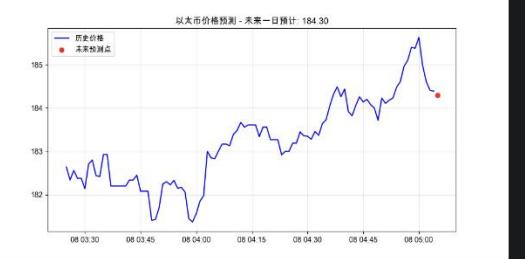
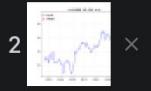
random forest.py

random forest forcast.py

```
43 next_date = last_date + timedelta(minutes=1) # 如果是分钟级的  
44  
45     print("-" * 30)  
46     print(f"预测结果: ")  
47     print(f"最后已知时间: {last_date} -> 价格: {prices[-1]}")  
48     print(f"预测未来时间: {next_date} -> 预计价格: {next_day_pred}")  
49     print("-" * 30)  
50  
51     # 6. 可视化最后一段走势和预测点  
52     plt.figure(figsize=(10, 5))  
53     plot_range = 100 # 只画最后100个点, 看得更清楚  
54     plt.plot(*args: df['datetime'].iloc[-plot_range:], prices[-plot_range:])  
55     plt.scatter(next_date, next_day_pred, color='red', label="未来预测点")  
56  
57     plt.title(f'以太币价格预测 - 未来一日预计: {next_day_pred}')  
58     plt.legend()  
59     plt.grid(visible=True, alpha=0.3)  
60     plt.show()  
61  
62 except Exception as e:  
63     print(f"运行中出现错误: {e}")
```

绘图

> 1,000x500 PNG (32位颜色) 42.26 kB



运行

random forest forcast

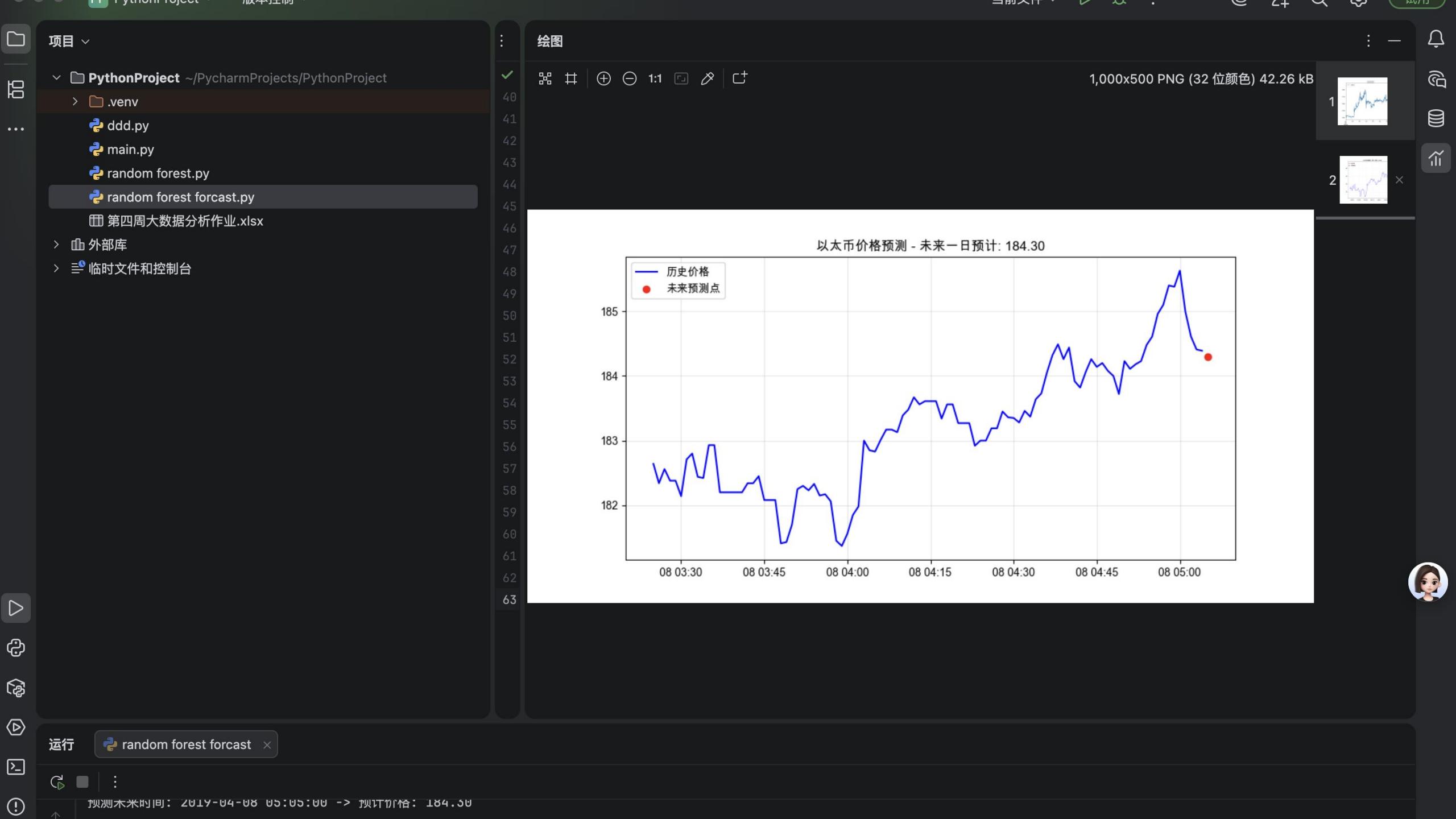


预测结果:

最后已知时间: 2019-04-08 05:04:00 -> 价格: 184.39

预测未来时间: 2019-04-08 05:05:00 -> 预计价格: 184.30







给我介绍下xgboost是什么

- 既然你正在用随机森林（Random Forest）做以太币价格预测，那么聊到 **XGBoost** 简直太精准了！

简单来说，如果随机森林是“靠人多力量大”的投票法，那么 **XGBoost** 就是“学霸带飞、不断纠错”的进化法。

- **XGBoost**

1. 什么是 **XGBoost**？

XGBoost 的全称是 **eXtreme Gradient Boosting**（极端梯度提升）。它不是一种全新的算法，而是基于 **GBDT**（梯度提升决策树）的一种高效实现方案。

在数据科学竞赛（如 Kaggle）中，XGBoost 被誉为“打表神器”，因为它处理结构化数据（就像你的 Excel 表格数据）的效果极其出色。

2. 核心逻辑：从“随机森林”到“**XGBoost**”

我们可以通过一个形象的类比来理解：

问问 Gemini 3

+ 工具

快速 ▾



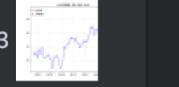
我们可以通过一个形象的类比来理解：

- **随机森林 (Random Forest) —— 平行投票：**你找了 100 个普通的分析师，让他们每人看一部分数据，各自给出一个以太币预测值。最后你取他们的**平均值**。如果有人看错了，其他人的正确结果能抵消错误。
- **XGBoost —— 迭代进化：**
 1. 你先让第 1 棵树去做预测，结果发现预测值比真实价格低了 100 刀。
 2. 第 2 棵树不再预测价格，而是专门去预测那缺失的 **100 刀误差**。
 3. 第 3 棵树再去预测前两棵树相加后剩下的微小误差。
 4. **每一棵新树都在为前人的错误“补课”**，最终加起来的结果非常接近真相。

3. XGBoost 为什么这么强？

它之所以被称为“极端”，是因为它在速度和精度上做了大量优化：

1. **正则化 (防止过拟合)**：它在算法里自带了“紧箍咒”，防止模型把噪音当成规律，这在波动的币圈数据中非常有用。
2. **并行处理**：虽然树是顺序生成的，但在寻找特征的分裂点时，它能利用 CPU 的多核并行计算，速度比普通提升树快得多。
3. **处理缺失值**：如果你的 Excel 里有空行，XGBoost 能自动学习缺失值的处理方向，不需要你手动填补。



项目

+ ⊕ ◁ × :

PythonProject ~/PycharmProjects/PythonProject

> .venv
ddd.py
main.py
random forest.py
random forest forcast.py
XGBoost.py

第四周大数据分析作业.xlsx

> 外部库

> 临时文件和控制台

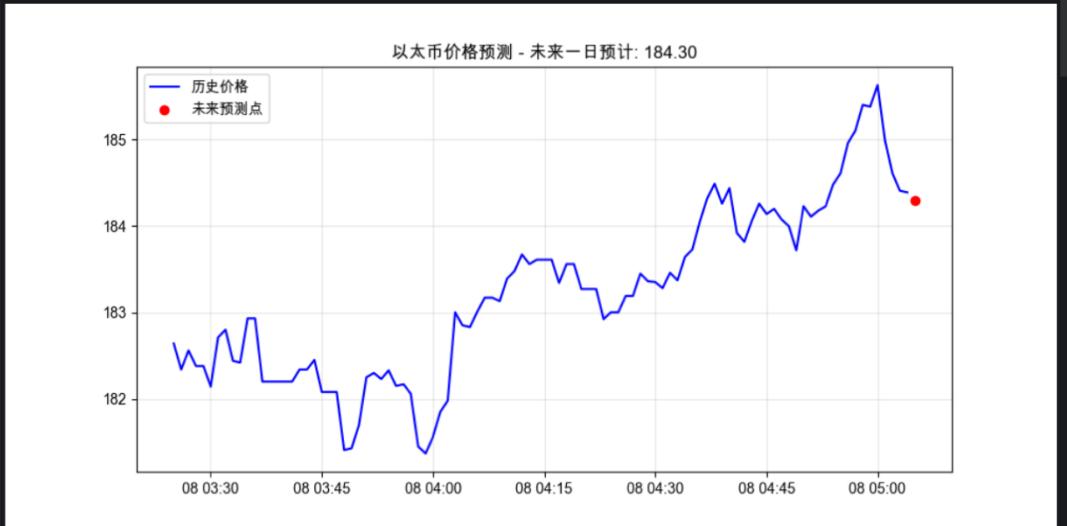
XGBoost.py

```
35     ^ 1 ~\n36     -行之后的一个点)\n37     点作为输入\n38     ow_size:].reshape(1\n39     ict(last_window)[0]\n40\n41\n42     .iloc[-1]\n43     nedelta(minutes=1)\n44\n45\n46     date} -> 价格: {pric\n47     date} -> 预计价格: {i\n48\n49     |\n50     )\n51     |100个点, 看得更清楚\n52     e'].iloc[-plot_rang\n53     t_day_pred, color='r\n54\n55     未来一日预计: {next_\n56\n57     :=0.3)\n58\n59     |\n60\n61     |\n62     |\n63     :}"|
```

绘图

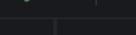
器 甘 ⊕ ⊖ 1:1 ↻ ⌂ ⊞

1,000x500 PNG (32位颜色) 42.26 kB



运行

XGBoost



为什么 XGBoost 和 Random Forest 的生成结果一样

1. 数据的“滞后性效应” (The Lag Effect)

在金融预测中，模型最稳妥的策略往往是“预测下一秒的价格和这一秒差不多”。

- **原因：**你的数据采样频率很高（分钟线），相邻两分钟的价格变动比例可能只有 **0.01%**。
- **结果：**无论是随机森林还是 XGBoost，它们都发现“预测明天价格 = 今天价格”产生的误差最小。因此，两个模型都倾向于给出一个非常接近当前值的预测，导致在图表上看起来两根线重合了。

2. 算法特征：它们本质上是“亲兄弟”

虽然 XGBoost 比随机森林更先进，但它们都属于 **集成决策树 (Tree Ensemble)** 算法：

- **随机森林 (RF)：**通过平行构建多棵树来降低方差。
- **XGBoost：**通过顺序构建多棵树来降低偏差。
- **共同点：**它们都无法处理超出训练集数值范围的数据。如果最近的价格一直平稳，两者的决策边界会非常相似，导致输出结果高度一致。

3. 特征过于单一

目前我们只喂给了模型一个特征：**历史价格 (Close)**。

- **现状：**这就好比给两个顶级厨师（XGBoost 和 RF）同样的食材（过去 20 分钟的价格），由于食材太单调，无论厨师技术再怎么不同，最后炒出来的菜味道都差不多。
- **解决建议：**如果加入 **成交量 (Volume)**、**均线 (MA)**、**RSI 指标** 等，XGBoost 捕捉非线性关系

食材太单调，无论厨师技术再怎么不同，最后炒出来的菜味道都差不多。

- 解决建议：如果加入 **成交量 (Volume)**、**均线 (MA)**、**RSI 指标** 等，XGBoost 捕捉非线性关系的能力就会展现出来，此时两者的预测图就会出现明显的差异。
-

如何让它们显示出“区别”？

如果你想看看这两个模型的真本事，可以尝试修改代码中的以下参数，观察它们的变化：

- 加大预测跨度：不要只预测未来 1 分钟，尝试预测未来 30 分钟。你会发现随机森林的预测线会趋于平缓（均值回归），而 XGBoost 可能会表现出更激进的趋势判断。
 - 调整学习率 (**learning_rate**)：在 XGBoost 代码里，尝试把 `learning_rate` 从 `0.03` 改成 `0.3`。你会发现 XGBoost 变得非常“敏感”，预测线会比随机森林抖动得更厉害。
 - 增加复杂特征：尝试把 `open`（开盘价）和 `high`（最高价）也放进 `X` 里。
-

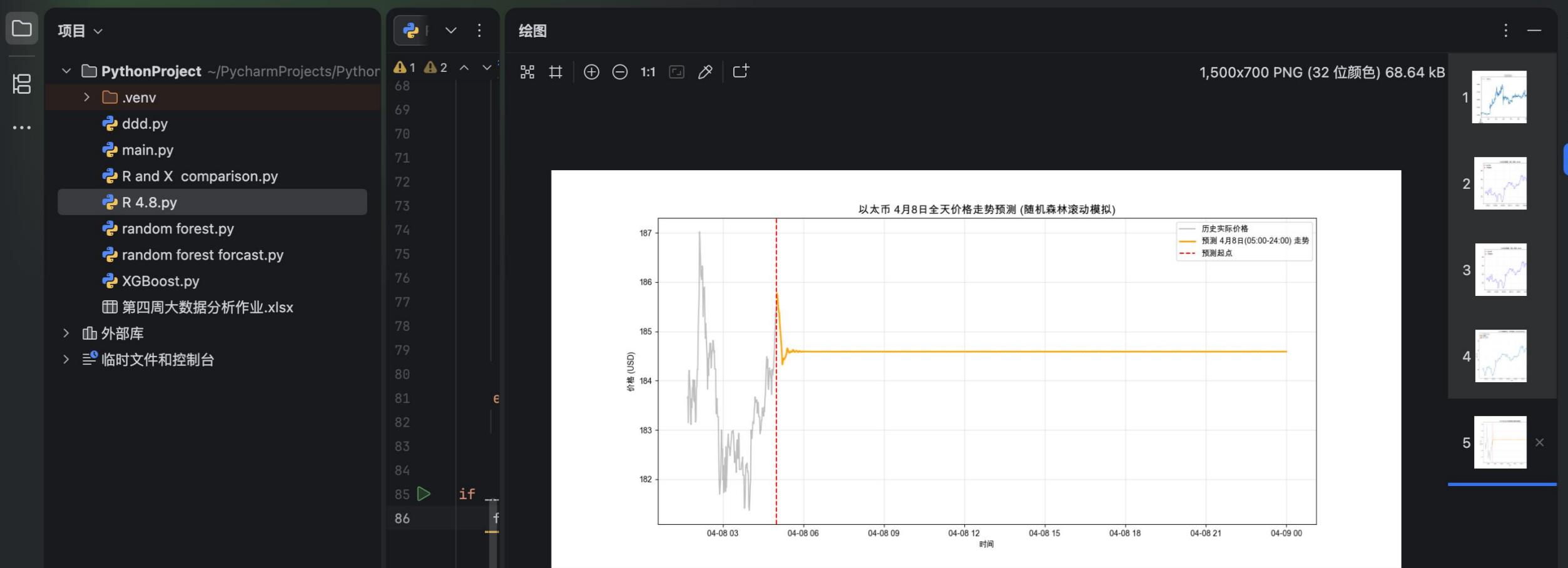
💡 总结建议

在目前这种简单的“单特征预测”下，**两图一样说明你的模型运行非常正常**，且两者都达到了当前特征下的最优预测。

你想让我帮你修改代码，一次性在同一张图上画出“真实价格”、“随机森林预测”和“XGBoost 预测”三根线进行对比吗？这样你可以更清晰地看到它们在转折点处的细微区别。

The screenshot shows a PyCharm interface with the following details:

- Project Structure:** PythonProject (~/PycharmProjects/Python) contains files: .venv, ddd.py, main.py, R and X comparison.py (selected), random forest.py, random forest forecast.py, XGBoost.py, and 第四周大数据分析作业.xlsx.
- Code Editor:** The R and X comparison.py file contains Python code for price forecasting using XGBoost. It includes imports for numpy, pandas, and xgboost, as well as plt for plotting. The code defines a run_comparison_forecast() function that plots actual data and XGBoost predictions.
- Run Output:** A plot titled "ETH 价格预测对比：多特征输入 (Close/Open/High)" is displayed. The y-axis is labeled "价格 (USD)" and ranges from 182 to 185. The x-axis is labeled "时间" (Time) and shows a timeline from 08:03:50 to 08:05:00. The plot compares "Actual" (blue line), "RF 预测点" (orange diamonds), and "XGBoost 预测点" (red triangles). The XGBoost model's predictions closely follow the actual price line.



Random Forest 预测全天失败 的原因

看到你发来的截图了！这正是随机森林在做长距离多步预测时最经典的现象，业界俗称“均值收敛”或“预测失效”。

即便模型运行成功了，预测线变成直线是由以下三个深层原因造成的：

1. 递归预测的“误差平滑”

代码中我们使用的是滚动预测（用预测出来的 $T + 1$ 去预测 $T + 2$ ）。

- **逻辑**：随机森林在本质上是在找历史数据的“平均值”。
- **结果**：第一步预测可能还有波动，但随着步数增加，模型由于看不到未来的真实输入，会变得越来越“保守”。它发现预测一个历史平均数比预测剧烈波动带来的误差更小，于是输出值迅速趋于稳定。

2. 缺乏“外部驱动力”

以太币的价格波动是由新闻、成交量、大户抛售等**外部变量**驱动的。

- **现状**：你的模型目前只知道“过去的价格”。
- **瓶颈**：当预测进行到几个小时后，由于没有新的外部信息（如成交量激增或暴跌）进入模型，随机森林就像一个失去方向的舵手，只能维持原地踏步，导致走势变平。

3. 决策树算法的局限性

项目

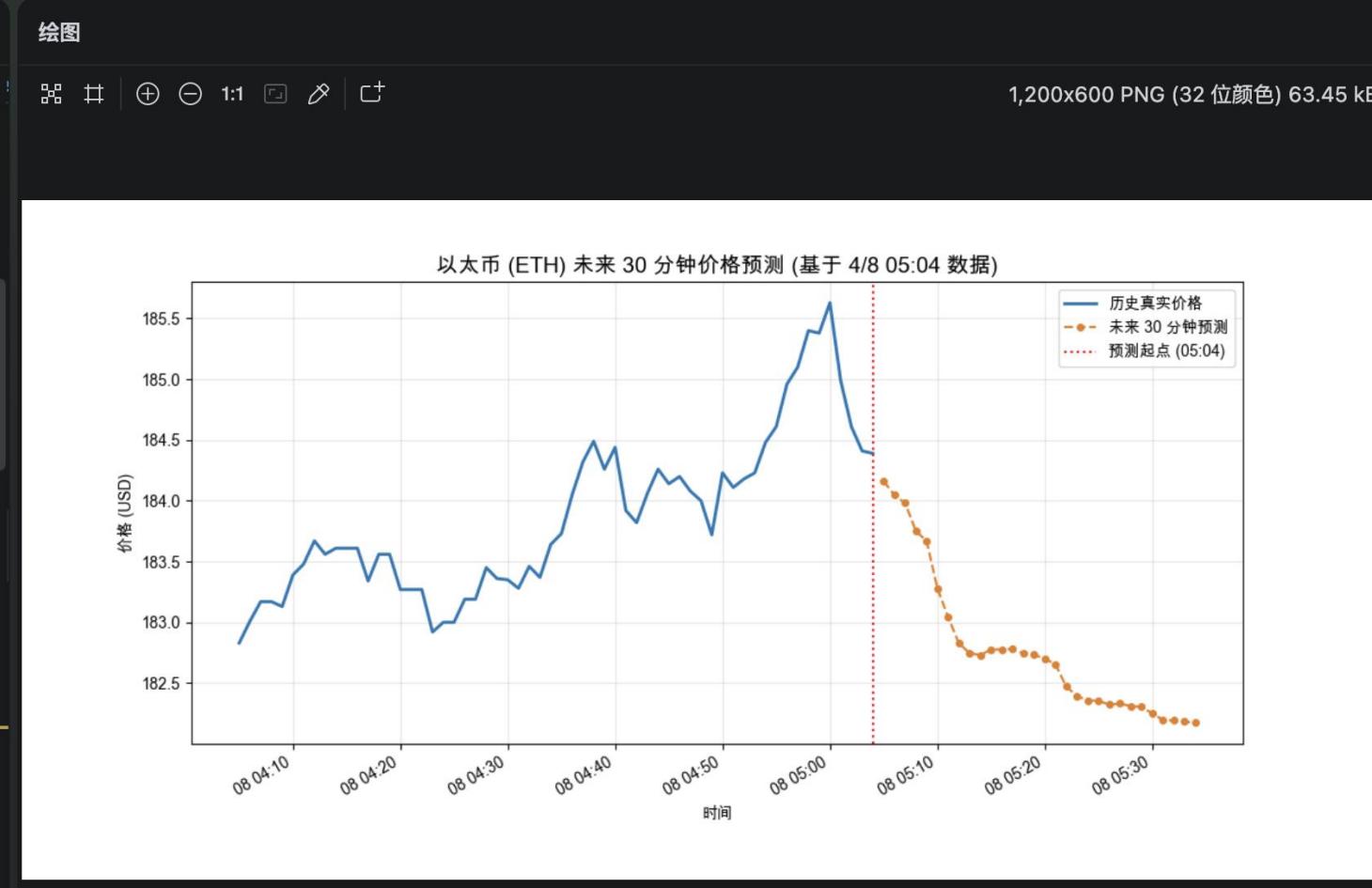
- PythonProject ~/PycharmProjects/Python
 > .venv
- ddd.py
- main.py
- R and X comparison.py
- R 4.8.py
- R 4.8 30min.py (当前)
- random forest.py
- random forest forecast.py
- XGBoost.py
- 第四周大数据分析作业.xlsx

> 外部库

> 临时文件和控制台

R 4.8

```
12 1 4 12
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```



运行 R 4.8 30min

C :

预测完成! 起点价格: 184.39
30分钟后预计价格: 182.17
预计总变动: -2.22

Random Forest 预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差0.51

项目

+ ⊕ ▲ × :

PythonProject ~/PycharmProjects/Python

> .venv

ddd.py

main.py

R and X comparison.py

R 4.8.py

R 4.8 30min.py

random forest.py

random forest forecast.py

XGB 4.8.py

XGBoost.py

第四周大数据分析作业.xlsx

> 外部库

> 临时文件和控制台

1 69
2 70
3 71
4 72
5 73
6 74
7 75
8 76
9 77
10 78
11 79
12 80
13 81
14 82
15 83
16 84
17 85
18 86
19 87
20 88
21 89
22 90
23 91



XGB预测4.8全天

运行 XGB 4.8 ×

：

```
/Users/mac/PycharmProjects/PythonProject/.venv/bin/python /Users/mac/PycharmProjects/PythonProject/XGB 4.8.py
```

✓ XGBoost 训练数据截断至: 2019-04-08 05:00:00

🚀 XGBoost 训练完成, 开始滚动预测全天走势...

进程已结束, 退出代码为 0



项目 ~

PythonProject ~/PycharmProjects/Python

- .venv
- ddd.py
- main.py
- R and X comparison.py
- R 4.8.py
- R 4.8 30min.py
- random forest.py
- random forest forecast.py
- XGB 4.8 30min.py
- XGB 4.8.py
- XGBoost.py

第四周大数据分析作业.xlsx

外部库

临时文件和控制台

XGB 4.8 30min.py

```
12 1 4 1 ^ v
88 label: '以太币'
89 '时间')
90 '价格 (USD)')
91 )
92 sible: True, a
93 utofmt_xdate
94
95 青
96 + " 预测简报"
97 | 价格: {train_
98 }钟后价格预测:
99 recast_price
100 |波动: {chang_
101 n as e:
102 运行失败: {e}
103
104
105
106
107 > aen_": orecast()
108
```



XGBoost预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差0.57

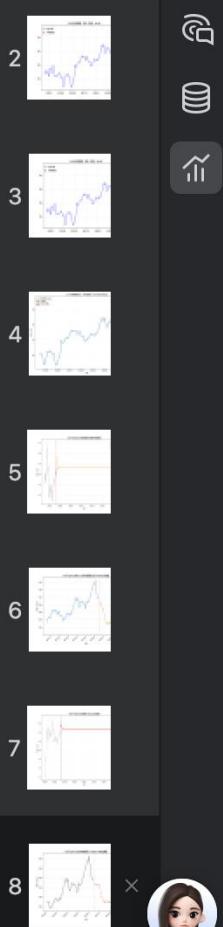
运行 XGB 4.8 30min

预测简报 -----

起点价格: 184.39

30分钟后价格预测: 183.25

预期波动: 1.14 (0.6172%)





lightbgm是什么



既然你已经玩过了 **Random Forest** 和 **XGBoost**, 那么聊到 **LightGBM** 就非常顺理成章了。它也是目前数据挖掘和金融预测领域最顶级的“三大神兵”之一（另外两个是 XGBoost 和 CatBoost）。

简单来说, **LightGBM (Light Gradient Boosting Machine)** 是由微软 (Microsoft) 开发的一种提升树模型。它的核心目标是: 比 **XGBoost** 更快, 占用内存更低, 且精度几乎不减。

1. 为什么叫 "Light" (轻量级) ?

在处理你的以太币数据 (假设数据量增加到几十万行) 时, XGBoost 可能会让电脑风扇狂转, 而 LightGBM 却能轻巧应对。它的“轻量”体现在两个核心黑科技:

- **直方图算法 (Histogram-based):** XGBoost 寻找分裂点时要遍历所有数值, 非常累。LightGBM 把连续的数值分到一个一个“桶”里 (直方图), 处理速度大幅提升。
- **GOSS 算法:** 它会自动保留那些误差较大的数据 (因为它们更有学习价值), 而随机过滤掉误差小的数据。这样模型计算量变小了, 但效果依然很好。

2. 核心区别：长高还是长胖？

这是 LightGBM 与其他模型最直观的区别：

- **XGBoost / 随机森林 (按层生长 - Level-wise)**: 像是在修剪一棵整齐的树，必须把这一层长满了，才去长下一层。这样比较稳，但有时会产生不必要的计算。
- **LightGBM (按叶子生长 - Leaf-wise)**: 它比较“任性”，哪里误差大（收益高），就往哪里深挖。这导致它的树可能长得“歪歪扭扭”，但能更快地降低误差。

3. LightGBM 的优缺点

特点	描述
速度极快	训练速度通常是 XGBoost 的数倍。
省内存	采用了直方图算法，内存消耗极低。
精度高	Leaf-wise 策略容易找到更深层的规律。
容易过拟合	注意！因为它喜欢“深挖”，如果数据量太小（比如只有几百行），它可能会过度解读噪音。



导出到 Google 表格





LightBGM结果的原因

1. 为什么随机森林（左图）最后是直线？

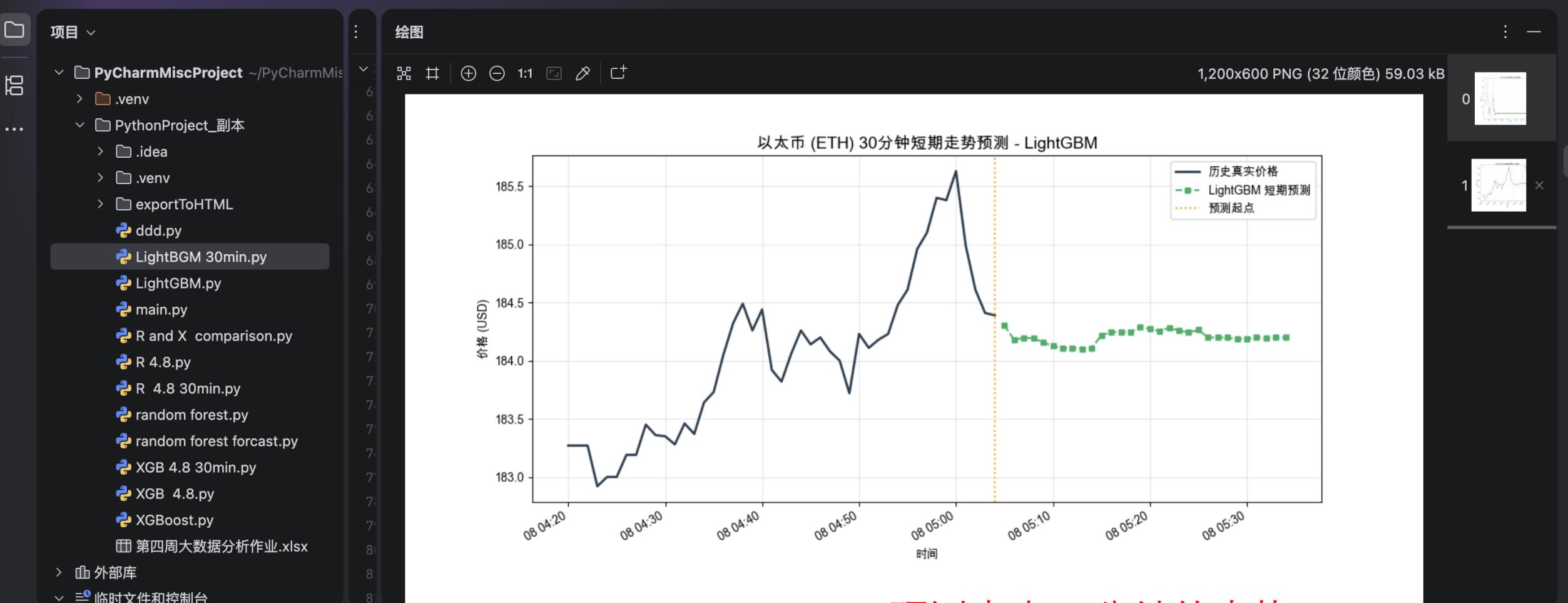
随机森林是由多棵独立的决策树组成的。

- **取平均值机制：**当它进行递归预测（用预测值推导预测值）到后期时，由于缺乏真实波动输入，每棵树给出的预测结果开始分歧。最终，模型选择了一个最保守的策略——输出所有结果的平均值。
- **结果：**就像一辆失去动力的车，最终慢慢滑行直到停在路边（均值），表现为一条毫无波动的水平线。

2. 为什么 LightGBM（右图）变成了“波浪线”？

这个结果非常有趣！LightGBM 的曲线没有变平，而是出现了规律性的锯齿状波动。

- **叶子生长（Leaf-wise）的副作用：**LightGBM 的决策树长得非常深且不对称，它对某些特定的数值极其敏感。
- **反馈回路（Feedback Loop）：**在滚动预测时，模型预测出的一个微小高值被放回窗口，导致下一步它认为该跌了；跌下去后，窗口特征又触发了它认为该涨的条件。
- **结果：**模型陷入了一个“自我循环”的怪圈。它不是在预测市场，而是在反复横跳。这在数学上叫“极限环”现象，代表模型已经失去了对真实趋势的判断，只是在机械地重复历史中某个微小的震荡模式。



LightGBM预测未来30分钟的走势图 30分钟后实际收盘价182.68 误差1.52



既然你已经熟悉了 XGBoost 和 LightGBM，那么 **CatBoost** (Categorical Boosting 的缩写) 就是你机器学习工具箱里的最后一块重要拼图。

它由俄罗斯搜索巨头 **Yandex** 在 2017 年开源，是目前处理**类别型特征** (Categorical Features) 最强的梯度提升模型。

1. 为什么它叫 CatBoost?

它的名字由来非常简单：**Categorical + Boosting**。在处理像“城市”、“币种”、“用户 ID”这类非数值型特征时，XGBoost 通常需要你手动做 One-Hot 编码，而 CatBoost 不需要任何预处理。你只需要告诉它哪些列是类别的，它会自动在内部完成更高质量的转换。

2. CatBoost 的三大“黑科技”

核心技术	说明	带来的好处
排序提升 (Ordered Boosting)	每次迭代都使用不同的随机排列数据，避免梯度偏差。	极强的抗过拟合能力，在小数据集上表现尤为出色。

用catboost模型，已知的4月1日到4月8日早上五点的数据，给我生成一个预测从4月8日5点零四分开始预测未来三十分钟的代码



既然你已经集齐了 XGBoost、LightGBM，现在加上 **CatBoost**，你的机器学习工具箱就彻底圆满了。

CatBoost 的**对称树（Symmetric Trees）**结构在处理这种高频波动数据时非常稳健，且它默认的过拟合保护机制非常适合这种递归式的短期预测。

🚀 CatBoost 未来 30 分钟走势预测代码

如果你还没有安装 CatBoost，请先在终端运行：`pip install catboost`

Python



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from catboost import CatBoostRegressor
```

项目 ✓

- <> R 4.8 30min.py.html
- <> random forest.py.html
- <> random forest forecast.py.html
- <> XGB 4.8 30min.py.html
- <> XGB 4.8.py.html
- <> XGBoost.py.html

new.venv

PythonProject_副本

- └ .idea
- └ .venv
- └ catboost_info
- └ exportToHTML
- ▶ 1.7.py
- ▶ catboost 4.8 30min.py
- ▶ ddd.py
- ▶ ETH.py
- └ ETH_USD_2019_04.xlsx
- ▶ LightBGM 30min.py
- ▶ main.py
- ▶ R and X comparison.py
- ▶ R 4.8.py
- └ R 4.8 30min.py

catboost ▾ ⋮

```
⚠ 1 ⚠ 4 ✅ 1 ^ v
pli
pli
pli
pli
pli
pli
pli
pli
pri

except
pri

if __name__ == "__main__":
    catboos
```

绘图



运行

运行 | catboost 4.8 30min ×



```
/Users/mac/PyCharmMiscProject/new.venv/bin/python /Users/mac/PyCharmMiscProject/PythonProject_副本/catboost 4.8 30min.py
```

CatBoost 正在分析历史规律 (4月1日 - 4月8日 05:00)

 CatBoost 正在训练中...

 预测完成! 起点价格: 184.39 -> 30分钟后预期价格: 184.07

进程已结束 退出代码为 0

Google Gemini

colab - 搜索

比特币价格预测方法

Untitled3.ipynb - Colab

https://colab.research.google.com/drive/1n4DPlxyFOe9O-wBQCxDi4EOxzpGZpj0#scrollTo=a623b8fc

文件 修改 视图 插入 代码执行程序 工具 帮助

命令 + 代码 + 文本 全部运行

RAM 磁盘

CO Untitled3.ipynb

[19] 0 秒

```
plt.grid(True, linestyle='--', alpha=0.7)
# Rotate x-axis labels for better visibility
plt.xticks(rotation=45)

# Adjust layout to prevent labels from being cut off
plt.tight_layout()

# Display the plot
plt.show()
```

Ethereum Price Prediction (Historical vs. Forecasted)

Final Task

Gemini

REMOVED. I've already corrected this in the code by removing the problematic line. The installed font should now be used correctly for the plot.

You're right, the Chinese characters in the plot title didn't render correctly despite the font installation. This is a common issue where Matplotlib's font cache needs to be refreshed. I've updated the code to clear Matplotlib's font cache, which should allow it to properly detect and use the installed Chinese font. This will ensure the title displays correctly.

读取第四周大数据分析作业.xlsx。用catboost模型，已知的4月1日到4月8日早上五点的数据，给我生成一个预测从4月8日5点零四开始预测未来三十分钟的代码，并运行，谢谢。

好的，我为您制定了一个计划。这个计划将加载数据、准备训练模型所需的特征，然后使用CatBoost模型预测从4月8日5点零四分开始。

显示 'df_future' 中 5:34 的预测数值

比较 'df_train_cleaned' 和 'df_future' 的 'close' 列

解释 CatBoost 模型训练的特征

需要我帮助您构建什么？

+ Gemini 2.5 Flash ▶

Gemini 的回答不一定正确，请注意核查，并谨慎使用代码。[了解详情](#)

Google Gemini

colab - 搜索

比特币价格预测方法

Untitled3.ipynb - Colab

https://colab.research.google.com/drive/1n4DPlxyFOe9O-wBQCxDi4EOxzpGZpj0#scrollTo=a623b8fc

Untitled3.ipynb

文件 修改 视图 插入 代码执行程序 工具 帮助

命令 + 代码 + 文本 全部运行

可视化程度低

秒

```
plt.grid(True, linestyle='--', alpha=0.7)
# Rotate x-axis labels for better visibility
plt.xticks(rotation=45)

# Adjust layout to prevent labels from being cut off
plt.tight_layout()

# Display the plot
plt.show()
```

Ethereum Price Prediction (Historical vs. Forecasted)

Price (USD)

Date

Final Task

Gemini

- Train CatBoost Model
- Generate Future Predictions
- Visualize Predictions
- Final Task

能不能给我把预测结果更清晰的显示，并且给我5点34分具体的数字。

Certainly! Here is a more detailed look at the future predictions. I'll display the entire `df_future` DataFrame, and then highlight the predicted price specifically for 5:34 AM.

You're right, the prediction for 05:34:00 AM was outside the previously defined range. I will now extend the prediction period to include 05:34:00 AM and regenerate the predictions.

抱歉，我遇到了错误，您能重试一下吗？

显示 `df_future` 中 5:34 的预测数值

比较 `df_train_cleaned` 和 `df_future` 的 `close` 列

解释 CatBoost 模型训练的特征

需要我帮助您构建什么？

+ Gemini 2.5 Flash ▶

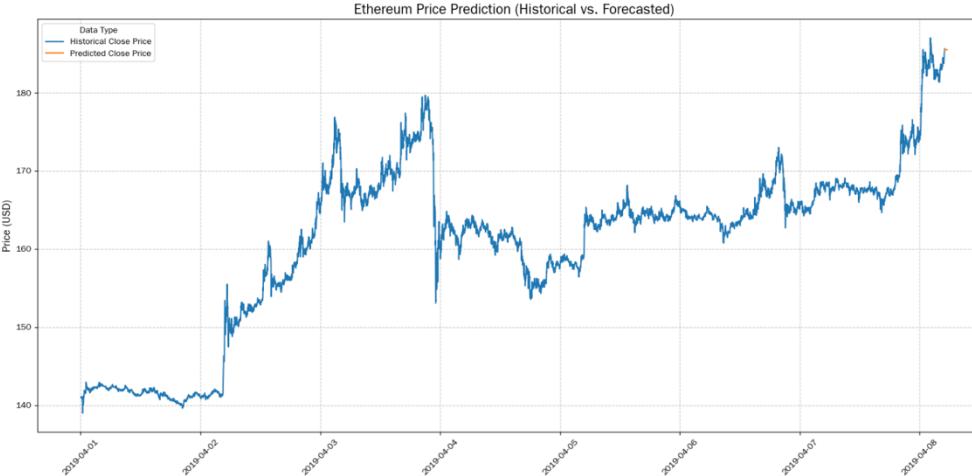
Gemini 的回答不一定正确，请注意核查，并谨慎使用代码。了解详情

The screenshot shows a Google Colab notebook titled "Untitled3.ipynb". The code cell contains plotting commands for an Ethereum price prediction chart. The chart, titled "Ethereum Price Prediction (Historical vs. Forecasted)", displays historical close prices in blue and forecasted close prices in orange. A red box highlights a specific data point on April 8th. To the right, a sidebar features a conversational interface with Gemini AI. The user asks for a clearer display of the prediction results and a specific value for 5:34 AM. Gemini responds by offering to show the entire future predictions DataFrame and highlights the specific 5:34 AM prediction. The user expresses a mistake about the time range and asks for a retry. The sidebar also includes links for viewing specific DataFrame rows, comparing columns, and explaining model features.

存在不运行的情况

```
[19] plt.grid(True, linestyle='--', alpha=0.7)  
# Rotate x-axis labels for better visibility  
plt.xticks(rotation=45)  
  
# Adjust layout to prevent labels from being cut off  
plt.tight_layout()  
  
# Display the plot  
plt.show()
```

Ethereum Price Prediction (Historical vs. Forecasted)



Gemini

- 正在生成...
- ✓ Preparing
- ✓ Load and Prepare Data
- ⌚ Train CatBoost Model
- ⌚ Generate Future Predictions
- ⌚ Visualize Predictions
- ⌚ Detailed Future Predictions
- ✓ Summary of Prediction Results and Insights

可可视化 df_train_cleaned 的 'close' 列

◆ Certainly! I will create a line plot to visualize the 'close' column of the `(df_train_cleaned)` DataFrame.

显示 `df_future` 中 5:34 的预测数值

比较 `df_train_cleaned` 和 `df_future` 的 'close' 列

解释 CatBoost 模型训练的特征

需要我帮助您构建什么?

+

Gemini 2.5 Flash ▶

Gemini 的回答不一定正确, 请注意核查, 并谨慎使用代码。了解详情

Random Forest 预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差0.51

XGBoost预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差0.57

Catboost预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差1.39

LightGBM预测未来30分钟的走势图
30分钟后实际收盘价182.68 误差1.52

30分钟模型预测准确度： Random Forest > XGBoost > Catboost> LightGBM

帮我写个代码画出4.8 all day.xlsx文件的走势图，然后帮我用catboost, random forest, xgboost和lightgbm用4月1日到4.8日早上五点的数据，预测未来十分钟的走势，在用这个十分钟的走势继续预测未来十分钟的走势，以此类推预测出4.8日全天的走势。对比4.8 all day的真实走势，给...

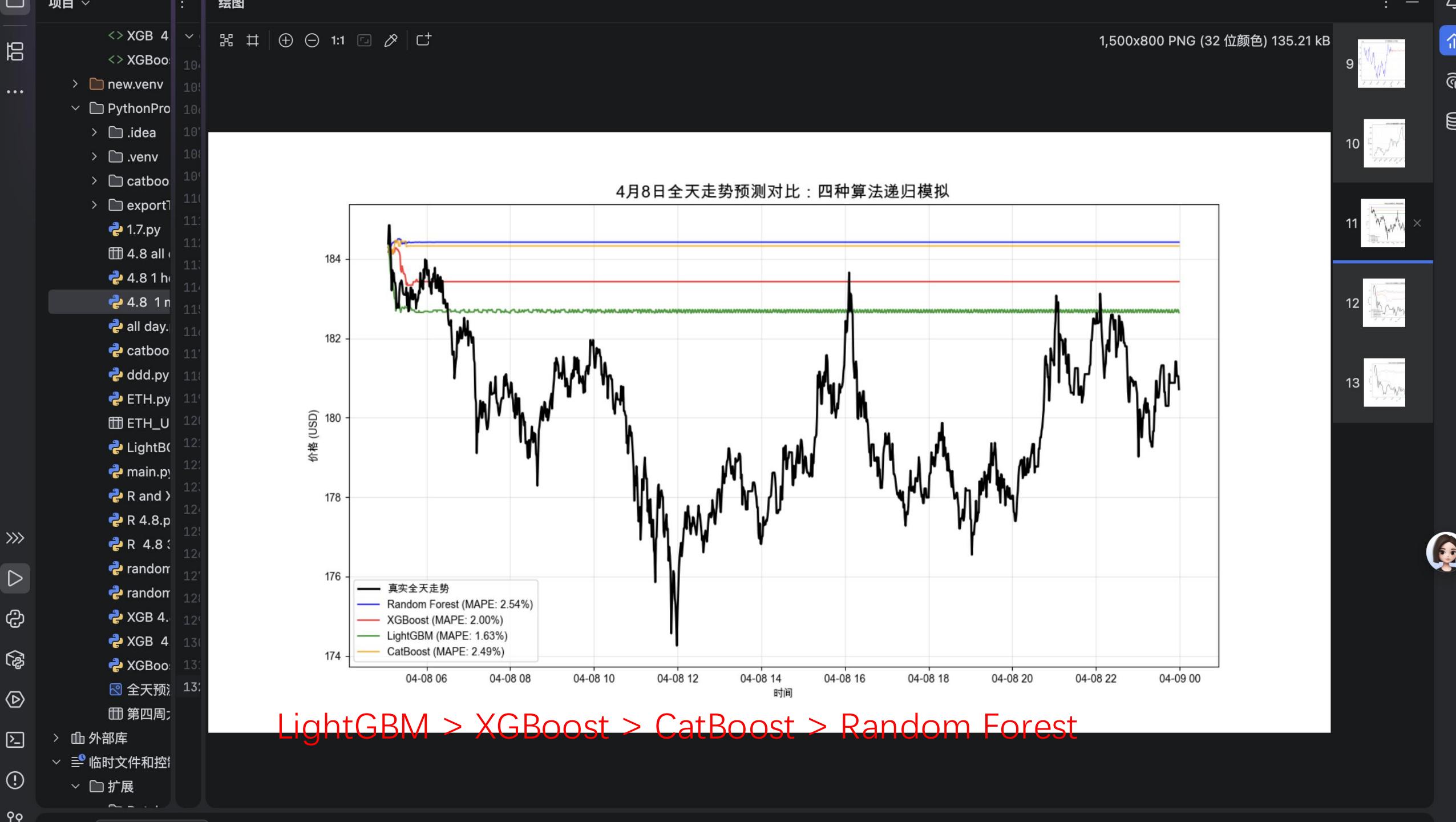


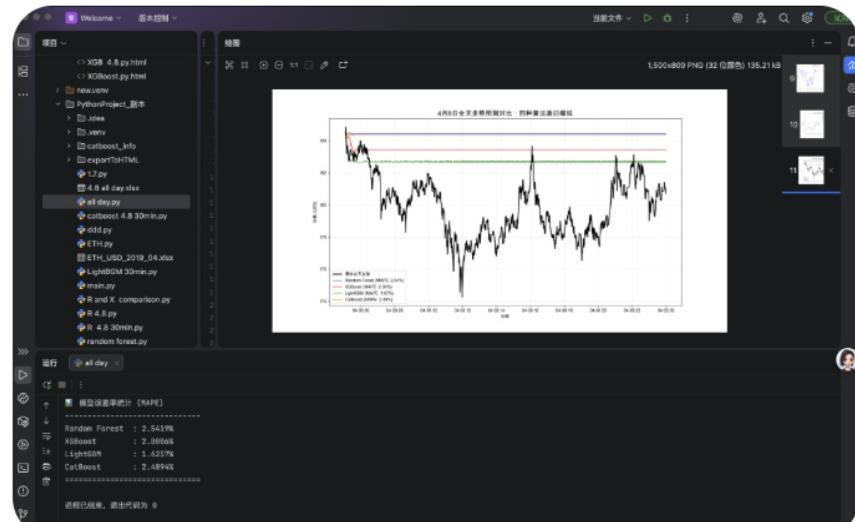
这是一个非常硬核的数据科学任务。你提到的这种预测方式在学术上叫“**多步递归滚动预测**”(**Multi-step Recursive Forecasting**)。

我们要先用已有的历史数据训练模型，然后让模型预测未来1分钟，再把这个预测值当作“真实发生”的数据喂回模型，去预测第2分钟，直到填满4月8日全天。

🛠 准备工作

确保你的虚拟环境安装了所有必要的库：





为什么后面四种模型都没显示趋势，你还能通过代码得出误差率

◆ 分析 ▼

看到你上传的第二张图，这正是**递归预测（Recursive Forecasting）**在金融时间序列任务中最典型、也最无奈的结果。

你问了一个非常深刻的问题：“为什么图上看着是直线（没趋势），代码却还能算出误差率？”



1. 为什么模型“不显示趋势”？

观察你的图表，四条预测线几乎都变成了水平直线。这并不是代码报错，而是模型在进行“长距离模拟”时集体丧失了动力。

- **均值回归陷阱：**当你让模型预测未来第 1 分钟时，它还能靠惯性给个坡度。但当你让它预测第 100 分钟、第 1000 分钟时，由于它是基于“自己预测的假数据”在推导，它无法预知未来的任何外部扰动（如大单买入、新闻）。
- **保守策略：**对于随机森林、XGBoost、LightGBM 这种决策树模型，当它们对遥远的未来完全不确定时，数学上的最优解就是输出训练集结果的**平均值**。
- **结果：**直线代表了模型给出的“安全答案”——也就是它认为未来价格最可能回落到的平稳区间。

2. 为什么能算出误差率？

误差率（MAPE）的计算公式是：

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{Actual_i - Forecast_i}{Actual_i} \right|$$



帮我写个代码画出4.8 all day.xlsx文件的走势图，然后帮我用catboost, random forest, xgboost和lightgbm用4月1日到4.8日早上五点的数据，预测未来十分钟的走势，在用这个十分钟的走势继续预测未来十分钟的走势，以此类推预测出4.8日6点04分的数据。对比4.8 all day的截止6点0...



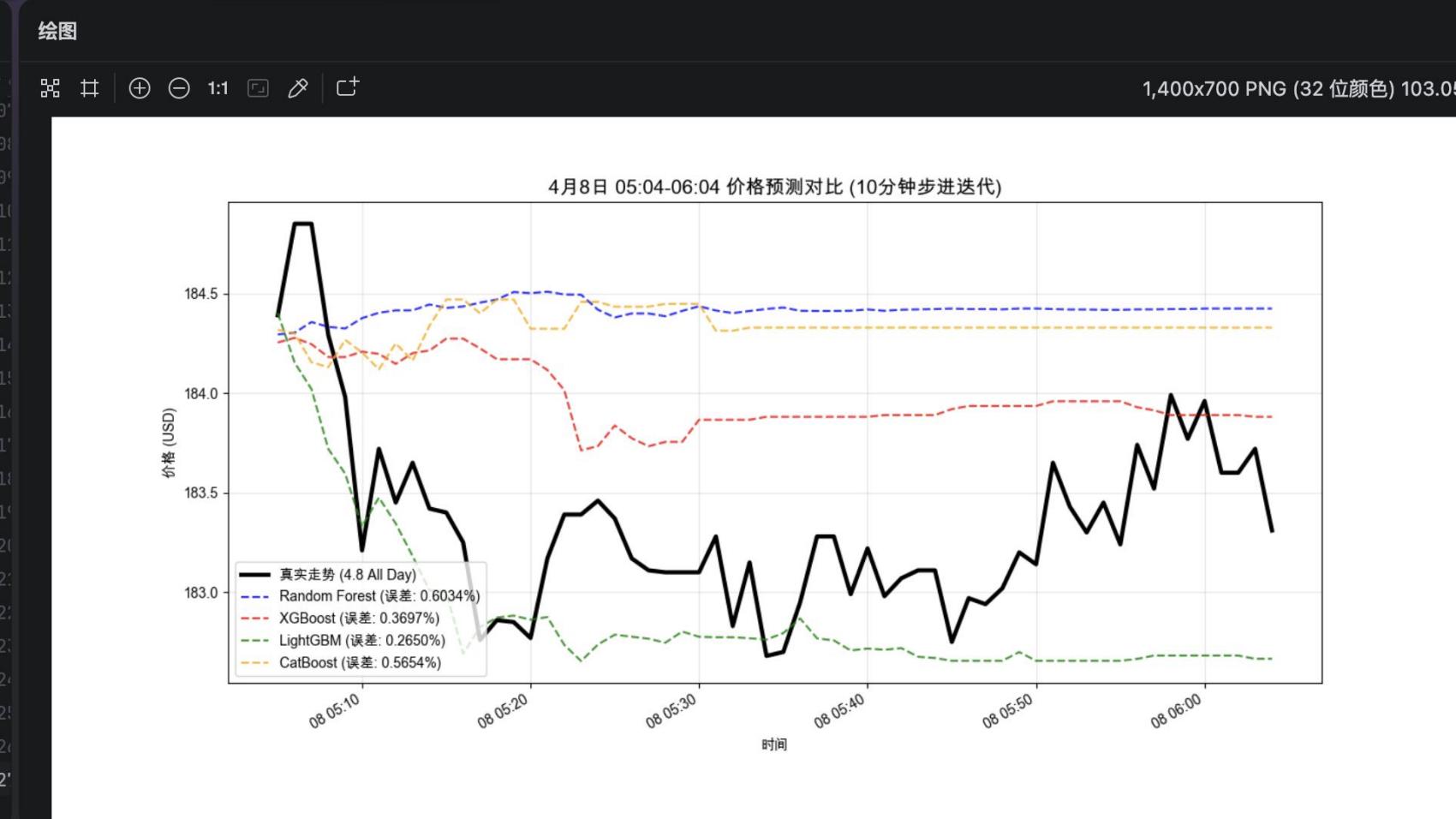
既然全天预测容易变成直线，我们这次调整策略，聚焦于从 **05:04 到 06:04** 这一个小时 的精细化预测。

这种“以 10 分钟为单位”的迭代预测，能让模型在短期内保持较好的动态感。我为你整合了四个模型并加入了误差计算逻辑。



项目

- <> XGB 4.8.py.html
- <> XGBoost.py.html
- > new.venv
- PythonProject_副本
 - > .idea
 - > .venv
 - > catboost_info
 - > exportToHTML
 - 1.7.py
 - 4.8 all day.xlsx
 - 4.8 1 hour.py
 - all day.py
 - catboost 4.8 30min.py
 - ddd.py
 - ETH.py
 - ETH_USD_2019_04.xlsx
 - LightBGM 30min.py
 - main.py
 - R and X comparison.py
 - R 4.8.py
 - R 4.8 30min.py

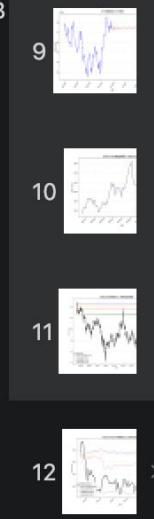


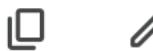
LightGBM > XGBoost > CatBoost > Random Forest

运行 4.8 1 hour

误差统计 (MAPE) - 截止 06:04

Random Forest	: 0.6034%
XGBoost	: 0.3697%
LightGBM	: 0.2650%
CatBoost	: 0.5654%





帮我写个代码画出4.8 all day.xls文件的走势图，然后帮我用catboost, random forest, xgboost和lightgbm用4月1日到4.8日早上五点的数据，预测未来1分钟的走势，在用这个1分钟的走势继续预测未来1分钟的走势，以此类推预测出4.8日6点04分的数据。对比4.8 all day的截止6点04分...



既然全天预测容易变成直线，我们这次将策略调整为最极致的逐分钟递归预测（**Recursive Point-to-Point**）。

代码将使用 4月1日至4月8日 05:04 的数据进行训练，然后每预测 1 分钟的价格，就立即将其作为特征喂回模型，连续迭代 60 次，直到对齐 4月8日 06:04 的真实数据。



四模型逐分钟迭代预测代码

Python



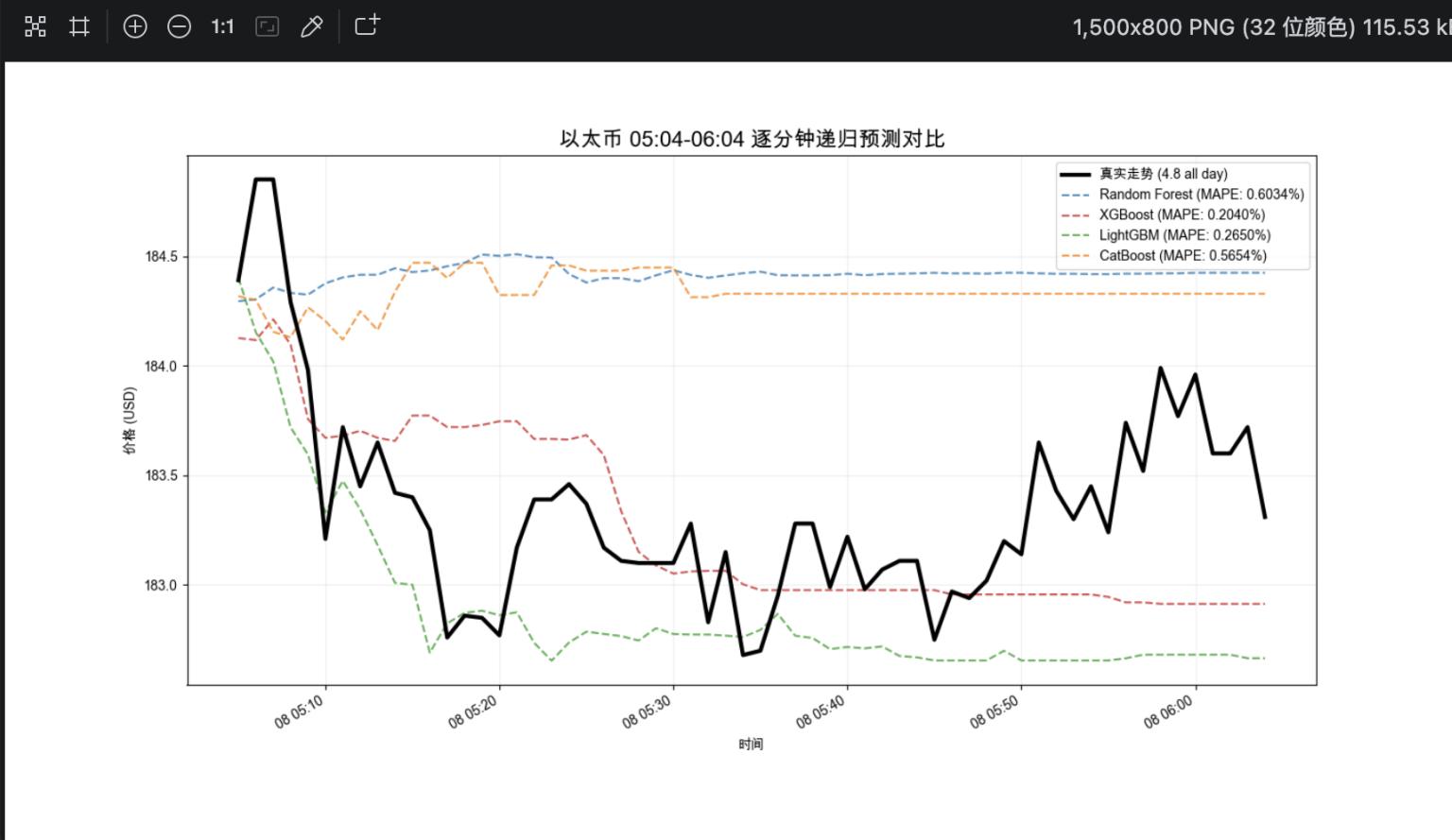
项目 ✓

- <> XGB 4.8.py.html
- <> XGBoost.py.html
- new.venv
- PythonProject_副本
- └ .idea
- └ .venv
- └ catboost_info
- └ exportToHTML
- └ 1.7.py
- └ 4.8 all day.xlsx
- └ 4.8 1 hour.py
- └ 4.8 1 min.py
- └ all day.py
- └ catboost 4.8 30n
- └ ddd.py
- └ ETH.py
- └ ETH_USD_2019_.xlsx
- └ LightBGM 30min.py
- └ main.py
- └ R and X_compari
- └ R 4.8.py

4.8 1 ⌂ ⌂

116	
117	
118	
119	阶段性误差
120	
121	
122	nape_sco
123	ed_score
124	{score
125	
126	
127	
128	')
129	
130	
131	▷ 
132	ion()

绘



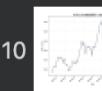
运行

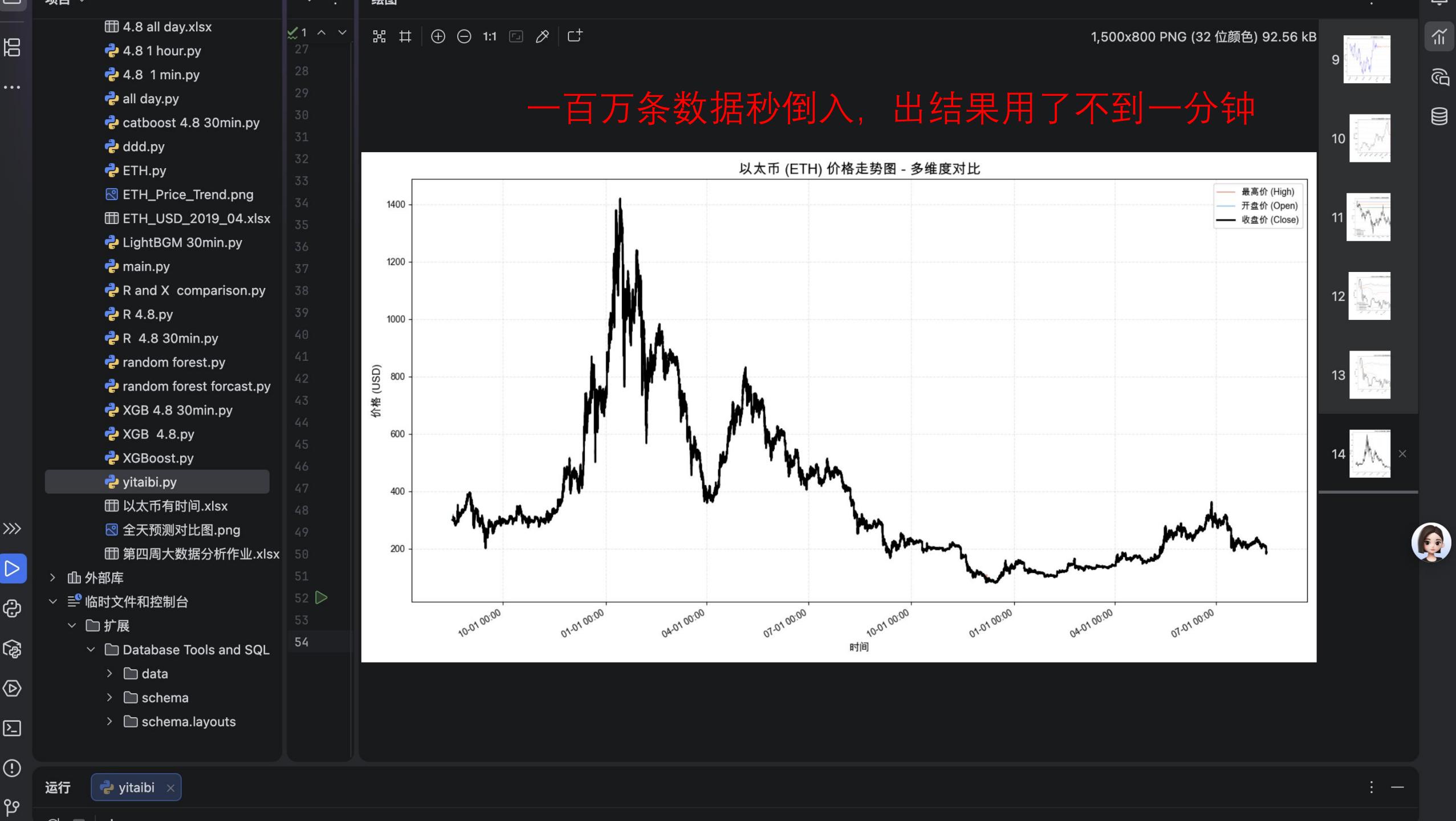
4.8 1 min ×

2

🏆 05:04-06:04 阶段性误差分析 (MAPE)

```
XGBoost      : 0.2040%  
LightGBM     : 0.2650%  
CatBoost     : 0.5654%  
Random Forest : 0.6034%
```

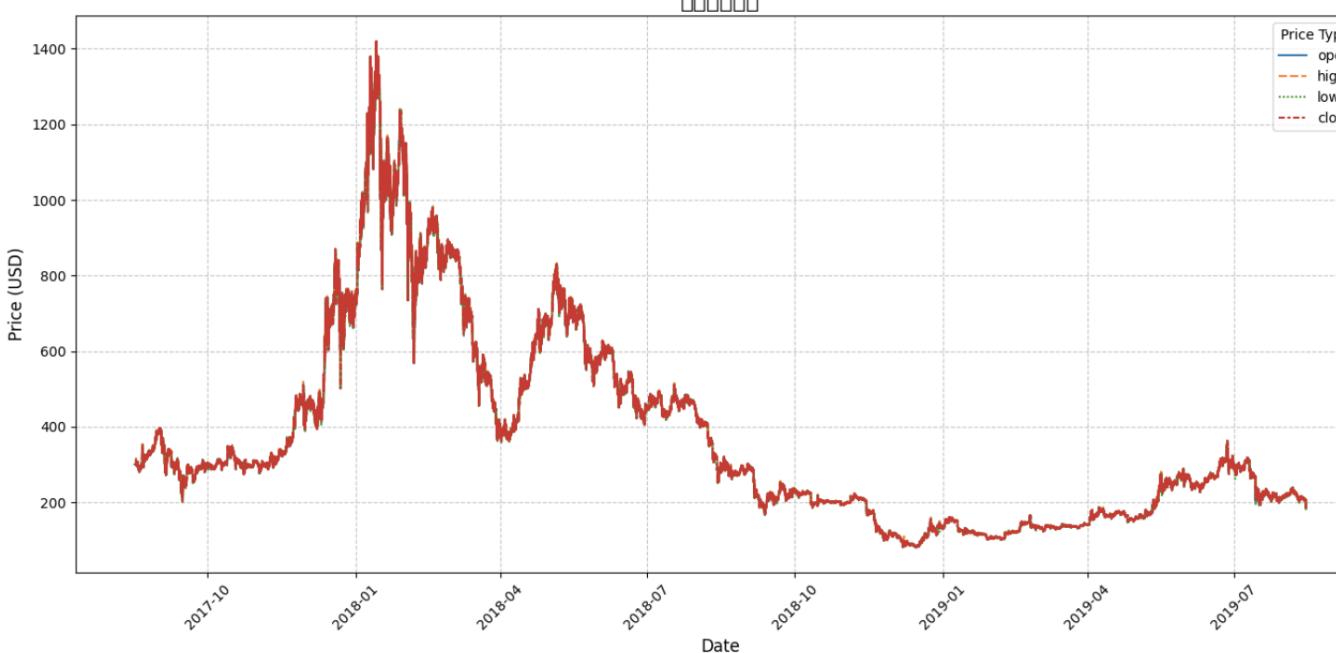




Colab导入读取三四分钟，出结果两分四十秒。

命令 + 代码 + 文本 | 全部运行 ▾

✓ RAM 磁盘



Gemini

 请解释以下错误：

④ `AttributeError: module 'matplotlib.font_manager' has no attribute 'FontProperties'`

- ◆ The error `AttributeError: module 'matplotlib.font_manager' has no attribute '_rebuild'` means that the `_rebuild()` function you were trying to use is no longer available in this version of Matplotlib. This function is an internal method and can change or be removed. I've already corrected this in the code by removing problematic line. The installed font should now be used correctly for the plot.

You're right, the Chinese characters in the plot title didn't render correctly despite the font installation. This is a common issue where Matplotlib's font cache needs to be refreshed. I've updated the code to clear Matplotlib's font cache, which should allow it to properly detect and use the installed Chinese font. This will ensure the title displays correctly.

▷ 接受并运行 ✓ 接受 ✘ 取消

需要我帮助您构建什么？

+

Gemini 2.5 Flash ▾

Gemini 的回答不一定正确, 请注意核查, 并谨慎使用代码。了解详情

python和colab网页版运行的优劣势

1. Google Colab (网页版)

Colab 是基于云端的 Jupyter Notebook 环境，非常适合快速实验和深度学习任务。

优势：

免费 GPU/TPU 资源：这是最大的杀手锏。对于运行 LSTM 或大型 CatBoost 模型，云端加速能比普通笔记本快几十倍。

零环境配置：预装了 pandas, numpy, scikit-learn, xgboost, tensorflow 等绝大多数库，不需要处理 pip install 报错。

易于分享：像 Google Docs 一样，一个链接就能把代码和运行结果（包括图表）发给别人。

跨设备：你可以在平板甚至手机上运行代码，计算都在云端。

劣势：

依赖网络：断网就无法编写或运行代码。

文件管理不便：**每次上传 .xlsx 文件后，如果会话断开，文件会被清空，必须挂载 Google Drive 或重新上传。**

运行时间限制：免费版连续运行超过 12 小时（或长时间无操作）会自动断开，不适合长达几天的训练任务。

2. 本地 Python (如 PyCharm / VS Code)

这是你目前正在使用的模式，更适合进行严肃的系统开发和本地数据处理。

优势：

本地文件交互：**读取本地的 4.8 all day.xlsx 非常方便，不需要反复上传。**

强大的调试 (Debug) 功能：PyCharm 的断点调试功能比 Colab 强大得多，能帮你快速定位代码错误。

无时间限制：只要你的电脑有电，代码可以跑一年。

安全性：数据存储在本地，不需要上传到云端，适合处理敏感隐私数据。

劣势：受限于硬件：如果你的电脑配置一般，跑复杂的随机森林或深度学习模型会非常卡顿。

环境冲突：正如你之前遇到的 ModuleNotFoundError 或 ImportError，需要自己管理虚拟环境 (.venv)。

代码管理非常方便

读取不同的数据表格也非常方便

The screenshot shows a Python development environment with the following details:

- Project Explorer:** Displays a folder named "PythonProject_副本" containing several subfolders and files. A red box highlights a group of files: "4.8 1 hour.py", "4.8 1 min.py", "all day.py", "catboost 4.8 30min.py", "ddd.py", "ETH.py", "ETH_Price_Trend.png", "ETH_USD_2019_04.xlsx", "LightBGM 30min.py", "main.py", "R and X comparison.py", "R 4.8.py", "R 4.8 30min.py", "random forest.py", "random forest forcast.py", "XGB 4.8 30min.py", "XGB 4.8.py", "XGBoost.py", "yitaibi.py", "以太币有时间.xlsx", "全天预测对比图.png", and "第四周大数据分析作业.xlsx".
- Code Editor:** Shows a Python script named "yitaibi.py" with the following code:

```
def plot_eth_trends(file_path):    1个用法
    plt.plot(*args: df['datetime'], df['open'], label='开盘价 (Open)')
    plt.plot(*args: df['datetime'], df['close'], label='收盘价 (Close)')

    # 5. 图表修饰
    plt.title(label='以太币 (ETH) 价格走势图 - 多维度对比', fontsize=14)
    plt.xlabel(xlabel='时间', fontsize=12)
    plt.ylabel(ylabel='价格 (USD)', fontsize=12)

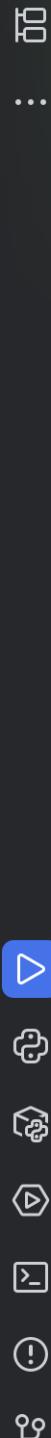
    # 优化时间轴显示
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.gcf().autofmt_xdate() # 自动倾斜日期标签防止重叠

    plt.legend(loc='best')
    plt.grid(visible=True, linestyle='--', alpha=0.3)

    # 6. 展示与保存
    plt.tight_layout()
    plt.savefig(*args: 'ETH_Price_Trend.png', dpi=300)
    print("✅ 绘图成功! 图片已保存为: ETH_Price_Trend.png")
    plt.show()

except Exception as e:
    print(f"❌ 运行失败, 错误信息: {e}")

if __name__ == "__main__":
    # 请确保文件名与你的文件完全一致
    plot_eth_trends("以太币有时间.xlsx")
```



```

> .venv
80
> catboost_info
81
> exportToHTML
82
    1.7.py
83
    4.8 all day.xlsx
84
    4.8 1 hour.py
85
    4.8 1 min.py
86
    all day.py
87
    catboost 4.8 30min
88
    ddd.py
89
    ETH.py
90
    ETH_Price_Trend.pr
91
    ETH_USD_2019_04
92
    LightBGM 30min.py
93
    main.py
94
    R and X comparison
95
    R 4.8.py
96
    R 4.8 30min.py
97
    random forest.py
98
    random forest forca
99
100
    XGBoost收益率分析
101
    XGB 4.8 30min.py
102
    XGB 4.8.py
103
    XGBoost.py
104
    yitaibi.py
105
    以太币有时间.xlsx
106
    全天预测对比图.png
107
    第四周大数据分析作
> 扩展

```

