بسم الله الرحمن الرحیم

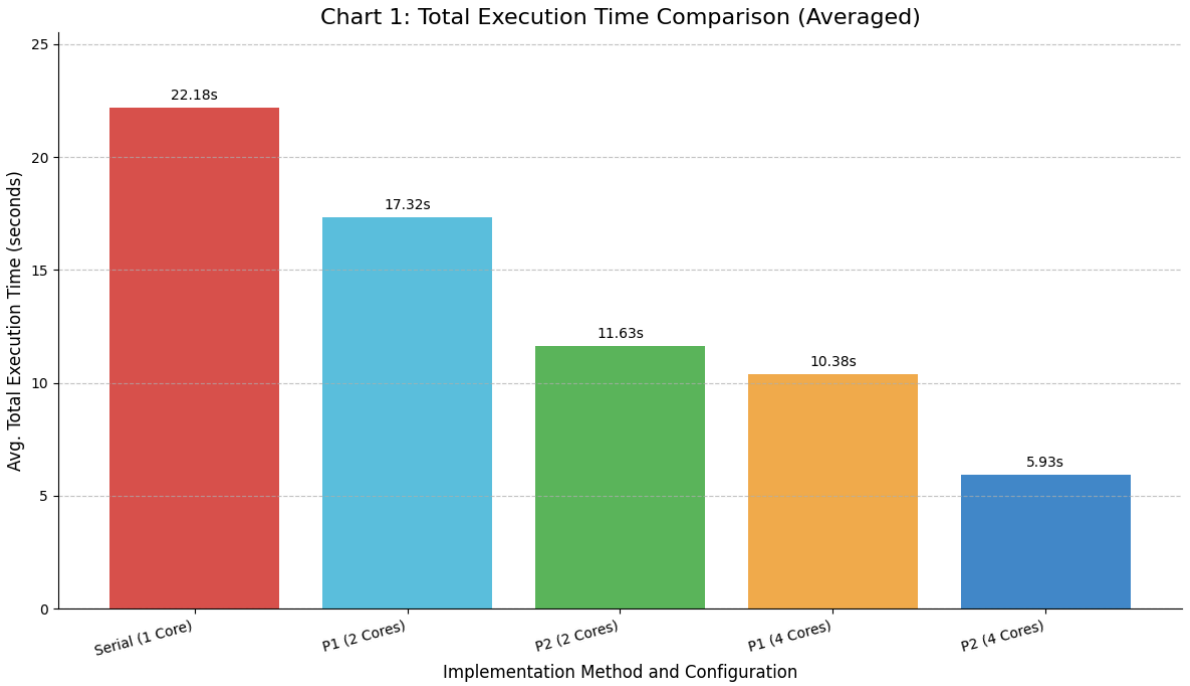
پروژه اول درس یادگیری ماشین توزیع شده

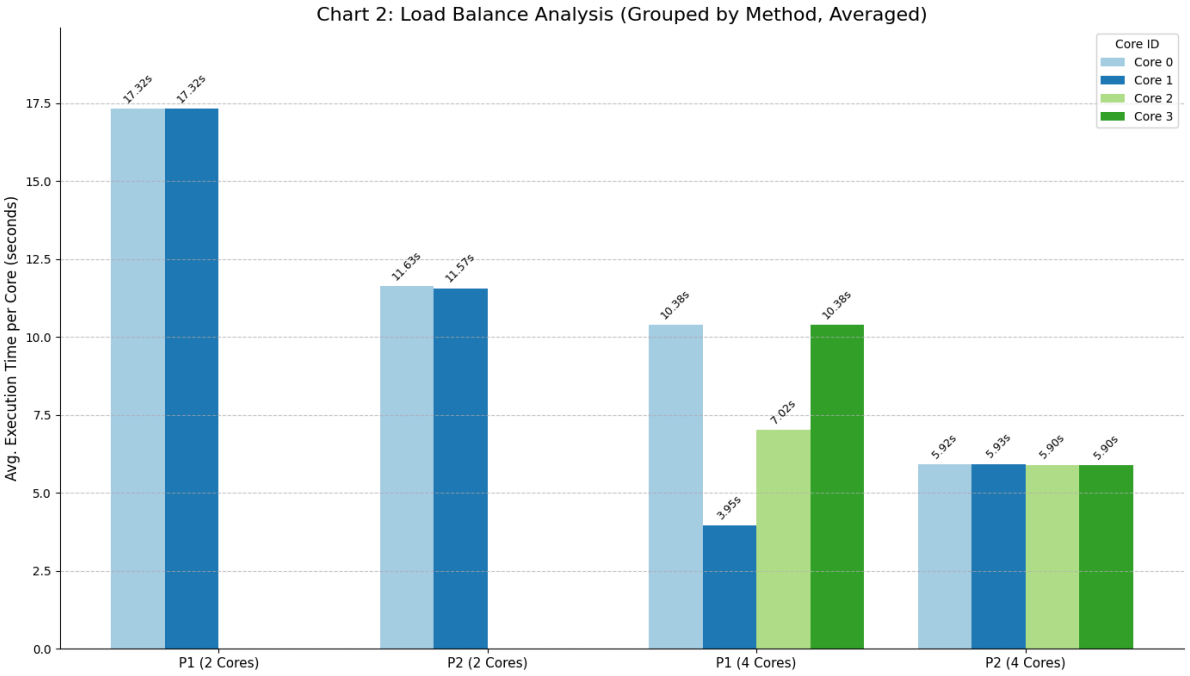
دکتر دوستی

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

فهرست

سوال ۱





همانطور که در نمودار های مشاهده می کنید، استفاده از ۲ هسته با روش اول زمان را در حدود ۲۲ درصد نسبت به سریال بهبود داده. یعنی ما با ۲ برابر منابع ۲۲ درصد بهبود داشتیم که دلیل آن عدم تقسیم متوازن وظایف است. این موضوع در تحلیل خود این حالت مشخص نیست زیرا پردازه اول منتظر می ماند نتیجه اولی را دریافت کند و سپس پردازش نهایی را انجام داده و خارج شود اما در حالت ۴ هسته ای می توانید تقسیم نامتوازن را به وضوح مشاهده کنید. در حالت دوم و در ۲ هسته نسبت به سریال ۴۷.۵ درصد بهبود

داشتیم که با توجه به ۲ برابر شدن منابع بسیار خوب است و ۲.۵ درصد که کم تر از حد انتظار بوده، به دلیل سربار تعامل است. در حالت ۲ گره و ۴ هسته در روش اول ۵۳.۲ درصد و در روش دوم ۷۳.۳ درصد بهبود داریم. مجدد در روش دوم بهبود بسیار مناسب است و ۱.۷ درصد سربار تعامل داریم. در روش اول اما همانطور که مشخص است کار ها به تناسب شکسته نشده و از منابع به خوبی استفاده نمی شود.
نکته ی دیگر این است که به خاطر تقریب های محاسباتی ای که در جمع و ضرب اعداد اعشاری وجود دارد و در هر کدام از این روش ها ترتیب متفاوت است ارقام کم ارزش عدد پی کمی باهم تفاوت دارند.

سوال ۲

بخش ۱

بر اساس دستورالعمل تمرین ، دو سناریو از نظر دقت و زمان اجرا مقایسه شده‌اند. (توجه: بر اساس فایل‌های اسکریپت و بخش ۲.۳ ، نرخ یادگیری (LR) در هر دو سناریوی عادی 0.01 در نظر گرفته شده است.)

داده‌های کلیدی

- **سناریوی ۱** (۱ دور ارتباطی، ۱۰ ایپاک محلی):
  - **دقت نهایی :** 0.6277
  - **کل زمان اجرا :** 15.22 ثانیه
  - **حداکثر زمان آموزش کلاینت:** 0.88 ثانیه (Rank 02)
- **سناریوی ۲** (۱۰ دور ارتباطی، ۱ ایپاک محلی):
  - **دقت نهایی :** 0.6358
  - **کل زمان اجرا :** 17.06 ثانیه
  - **حداکثر زمان آموزش کلاینت (در مجموع ۱۰ دور):** ~0.29 ثانیه (Rank 02)

تحلیل مقایسه ای

- **دقت :** سناریوی ۲ (ارتباطات بیشتر) به دقت نهایی **بالاتری** (0.6358) نسبت به سناریوی ۱ (محاسبات محلی بیشتر) (0.6277) دست یافت.
- **زمان :** سناریوی ۱ (ارتباطات کمتر) با زمان ۱۵.۲۲ ثانیه، **سریع‌تر** از سناریوی ۲ (۱۷.۰۶ ثانیه) اجرا شد. اگرچه زمان *محاسبات محلی* در سناریوی ۱ به طور قابل توجهی بیشتر بود (حداکثر 0.88 ثانیه در مقابل 0.29 ثانیه)، اما سناریوی ۲ متحمل هزینه سربار ناشی از ۹ دور ارتباطی اضافه (شامل ارسال، دریافت و همگام‌سازی) شد که این هزینه سربار ارتباطی، از زمان صرفه‌جویی شده در محاسبات بیشتر بود.

هزینه و فایده

این نتایج به خوبی تبادل کلاسیک بین محاسبات محلی و ارتباطات در یادگیری فدرال را نشان می‌دهند:

- **افزایش محاسبات محلی (مانند سناریوی ۱):**
  - **فایده:** سربار ارتباطی را به شدت کاهش می‌دهد، زیرا کلاینت‌ها فقط یک بار مدل‌ها را ارسال و دریافت می‌کنند. این امر منجر به **کاهش زمان اجرای کل** شد (۱۵.۲۲ ثانیه).

- **هزینه:** وقتی کلاینت‌ها برای ایپاک‌های زیاد (۱۰ ایپاک) به صورت محلی آموزش می‌بینند، مدل‌های آن‌ها شروع به واگرایی از یکدیگر و همگرایی به سمت داده‌های محلی خود (که ناهمگون یا Non-IID هستند) می‌کنند. هنگامی که سرور در انتها این مدل‌های واگرا شده را میانگین‌گیری می‌کند، مدل جهانی حاصل ممکن است بهینه نباشد. این پدیده منجر به **دقت نهایی پایین‌تر** (0.6277) شد.

- **افزایش ارتباطات (مانند سناریوی ۲):**
  - **فایده:** کلاینت‌ها به طور مکرر (پس از هر ۱ ایپاک) با سرور همگام می‌شوند. این همگام‌سازی مکرر، مدل‌های محلی را نزدیک به میانگین جهانی نگه می‌دارد و از واگرایی شدید آن‌ها جلوگیری می‌کند. این فرآیند به مدل جهانی اجازه می‌دهد تا به راه‌حل بهتری که برای *تمام* داده‌ها مناسب است همگرا شود، که منجر به **دقت نهایی بالاتر** (0.6358) گردید.
  - **هزینه:** هر دور ارتباطی هزینه زمانی قابل توجهی دارد. تکرار این هزینه برای ۱۰ دور، زمان اجرای کل را **افزایش** داد (۱۷.۰۶ ثانیه).

بخش ۲

بر اساس دستورالعمل بخش ۲.۳ ، سناریوی دوم (۱۰ دور، ۱ ایپاک) مجددا اجرا شد، با این تفاوت که کلاینت ۱ (Rank 01) از نرخ یادگیری مخرب 0.5 استفاده کرد، در حالی که سایر کلاینت‌ها (Ranks 02, 03) از نرخ یادگیری عادی 0.01 استفاده کردند.

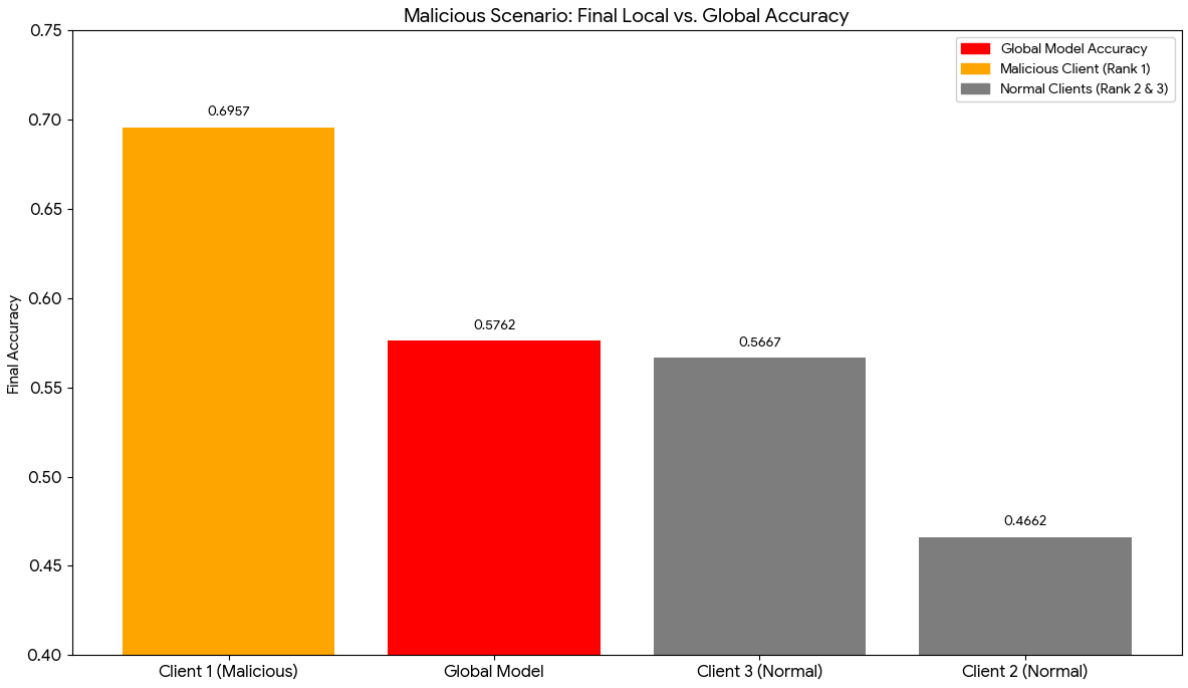
داده‌های کلیدی

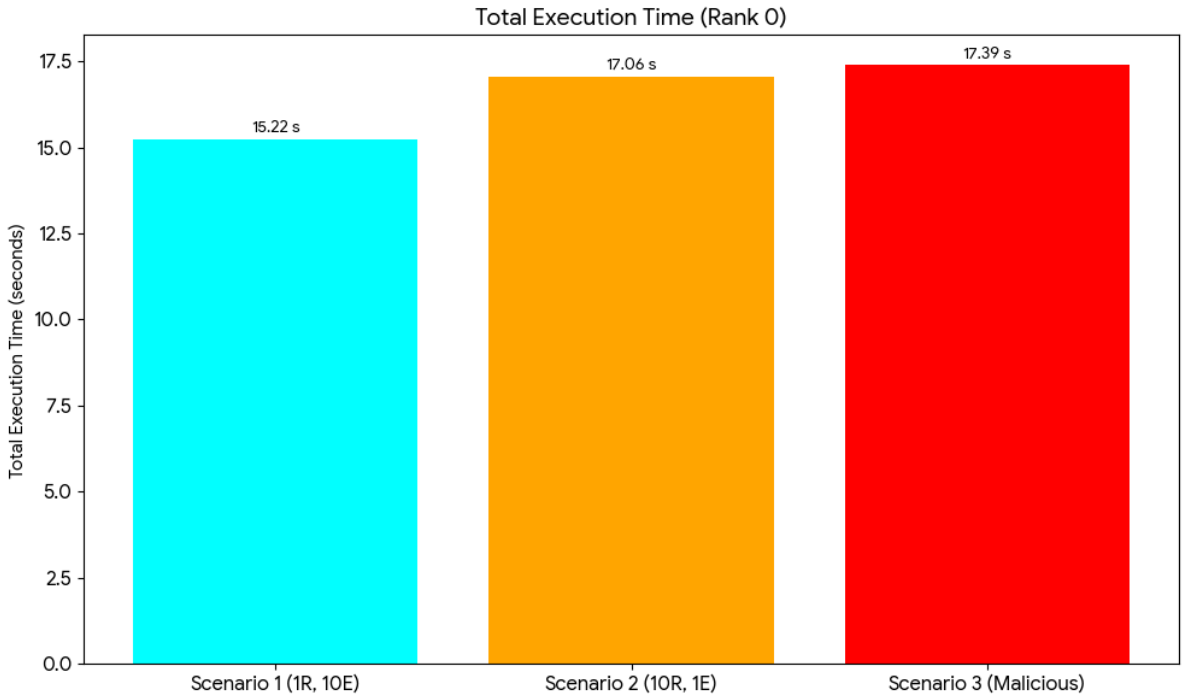
- **سناریوی ۲ (عادی - LR=0.01):**
  - **دقت نهایی :** 0.6358
- **سناریوی مخرب (LR_R1=0.5, LR_R2/3=0.01):**
  - **دقت نهایی :** 0.5762
  - **دقت محلی کلاینت‌ها:**
    - Rank 01 (مخرب): 0.6957 (بسیار بالا)
    - Rank 02 (عادی): 0.4662 (بسیار پایین)
    - Rank 03 (عادی): 0.5667 (پایین)

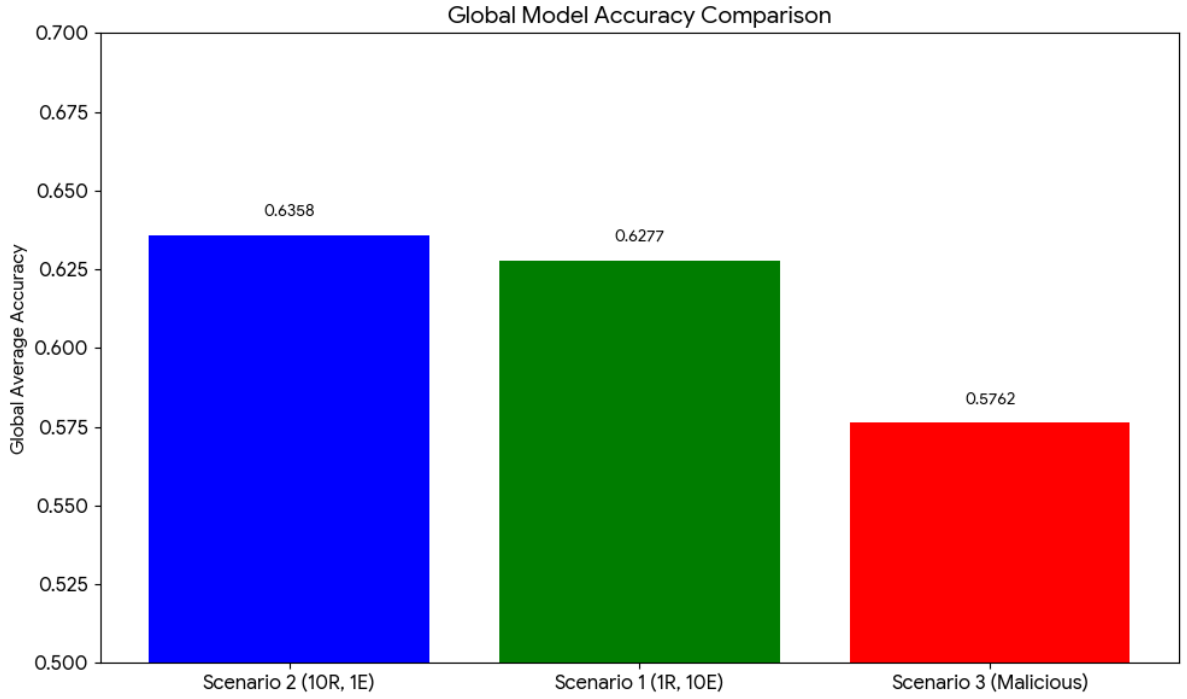
تحلیل نتایج

1. **تأثیر بر دقت سراسری:** نرخ یادگیری بسیار بالای کلاینت مخرب باعث **کاهش چشمگیر** دقت مدل سراسری نهایی شد؛ دقت از 0.6358 در حالت عادی به 0.5762 در حالت مخرب سقوط کرد.
2. **دلیل پدیده :**

- نرخ یادگیری (LR) اندازه گام‌هایی را که مدل هنگام بهینه‌سازی وزن‌های خود برمی‌دارد، کنترل می‌کند. نرخ یادگیری عادی (0.01) گام‌های کوچک و پایداری را تضمین می‌کند.
- نرخ یادگیری 0.5 به طور افراطی بزرگ است. هنگامی که کلاینت مخرب (Rank 01) آموزش می‌بیند، به جای همگرایی آرام، گام‌های بسیار بزرگی برمی‌دارد. این گام‌ها باعث می‌شوند وزن‌های مدل به شدت از نقطه بهینه "پرتاب" (Overshoot) شوند و به منطقه‌ای نامناسب در فضای پارامترها واگرا گردند.
- در پایان هر دور، سرور وزن‌های تمام کلاینت‌ها را میانگین‌گیری می‌کند. وزن‌های "آلوده" و واگرا شده از Rank 01 با وزن‌های "سالم" از Ranks 02 و 03 ترکیب می‌شوند.
- این فرآیند میانگین‌گیری، مدل سراسری را از مسیر بهینه خود خارج کرده و آن را به سمت وزن‌های نامناسب کلاینت مخرب "می‌کشد".
- همانطور که در خروجی malicious.out دیده می‌شود، بایاس (bias) مدل جهانی به سرعت به مقادیر بسیار منفی (تا -0.54) رانده می‌شود، در حالی که در اجرای عادی (scenario_2.out) در مقادیر بسیار کوچک (تا -0.0003) باقی می‌ماند.
- در نتیجه، مدل جهانی نهایی، یک مصالحه نامناسب بین مدل‌های سالم و مدل مخرب است که منجر به عملکرد ضعیف (دقت 0.5762) بر روی داده‌های تست کلاینت‌های عادی می‌شود. (جالب اینکه خود کلاینت مخرب به دلیل Overfit شدید روی داده‌های محلی خود با LR بالا، دقت محلی بالایی را گزارش می‌دهد).



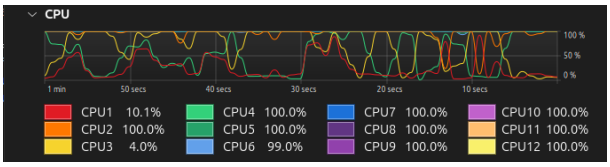

Total Execution Time (Rank 0)
Total Execution Time (seconds)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
15.22 s
17.06 s
17.39 s
Scenario 1 (1R, 10E)
Scenario 2 (10R, 1E)
Scenario 3 (Malicious)

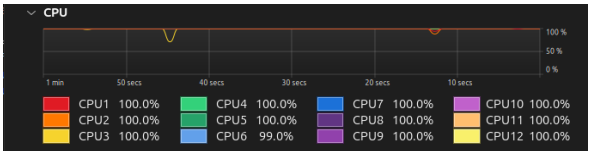

Global Model Accuracy Comparison
Global Average Accuracy
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.6358
0.6277
0.5762
Scenario 2 (10R, 1E)
Scenario 1 (1R, 10E)
Scenario 3 (Malicious)

سوال ۳

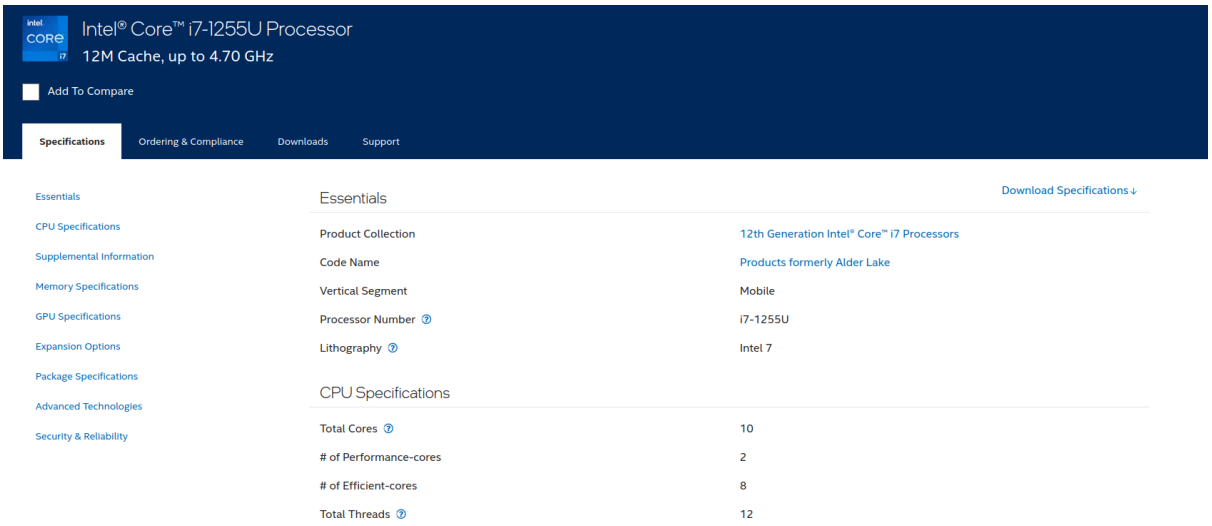
```
benchmark with seed 13847
time    for det         10.0k*10.0k     with scipy-openblas:    7.54
time    for det         10.0k*10.0k     with mkl-sdl:           7.63
time    for det         20.0k*20.0k     with scipy-openblas:    60.12
time    for det         20.0k*20.0k     with mkl-sdl:           59.42
time    for inv         10.0k*10.0k     with scipy-openblas:    27.86
time    for inv         10.0k*10.0k     with mkl-sdl:           26.54
time    for inv         20.0k*20.0k     with scipy-openblas:    214.22
time    for inv         20.0k*20.0k     with mkl-sdl:           198.84
```

نتایج به شکل بالا است و همانطور که مشاهده می کنید بهبود چیزی کمتر از ۵ درصد است اما در هنگام اجرا تست یک مورد عجیب مشاهده شد. (تصویر سمت چپ برای mkl و تصویر راست برای open blas)





در mkl به طرز عجیبی از ۲ رشته پردازنده استفاده نمی شود. یعنی عملا با ۱۰ رشته کمی بهتر از ۱۲ رشته open blas عمل کرده یعنی حدود ۱۶ درصد منابع کمتری مصرف کرده. دلیل این موضوع احتمالا مشکلات پیکربندی بوده و نصب mkl با دردسر های زیادی همراه بود.



دلیل احتمالی دیگر هم این است که پردازنده من ۸ هسته کم مصرف تک رشته دارد و ۲ هسته پرمصرف ۲ رشته و ۲ رشته ای که کار نمی کند می تواند همان ۲ رشته ای باشند که هسته مستقل ندارند.

منابع

https://gemini.google.com/share/f85a3b11bda3
https://gemini.google.com/share/c12c82694523
PyTorch. Logistic Regression is a fundamental… | by Carlos Rodrigo Coelho | Medium
https://gemini.google.com/share/0a173bd9bc9f
https://gemini.google.com/share/7123b74ae89c
https://gemini.google.com/share/9b4bb5388b28