Optiver 参戦記 & 銀メダル解法

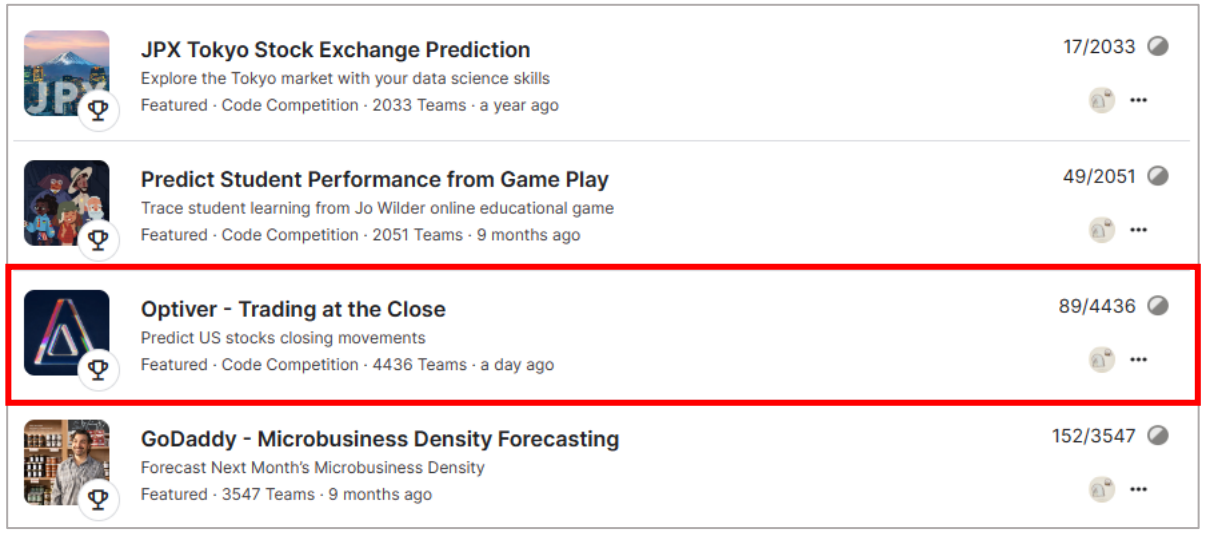
tonic

2024/05/23

1. 自己紹介

tonic（@tonic3561）

▶ フリーランスDS
▶ Kaggler



目次

1. ざっくり解法
2. 思考と試行の時系列振り返り
    a. 問題理解
    b. ブレスト & インプット
    c. ベースライン作成
    d. ひたすら実験
3. 上位解法との差分考察

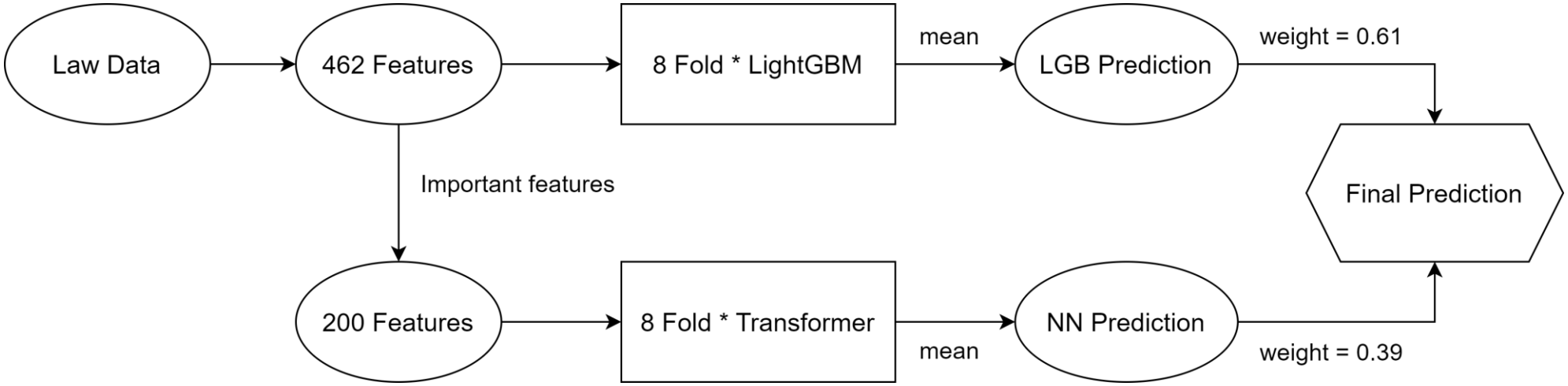
目次

1. ざっくり解法
2. 思考と試行の時系列振り返り
    a. 問題理解
    b. ブレスト & インプット
    c. ベースライン作成
    d. ひたすら実験
3. 上位解法との差分考察

1. ざっくり解法

- LightGBM と Transformer のアンサンブル
  - 特徴量数: 462
  - 8Fold CV の全モデルの予測値を平均
  - LGB : NN = 0.61 : 0.39 でブレンド

※ 解法詳細はこちら



目次

1. ざっくり解法
2. 思考と試行の時系列振り返り
    a. 問題理解
    b. ブレスト & インプット
    c. ベースライン作成
    d. ひたすら実験
3. 上位解法との差分考察

2. 時系列振り返り

■ tonic流 いつもの機械学習タスクの倒し方

1. 問題理解 （0.5週間）
2. ブレスト & インプット（0.5週間）
3. ベースライン作成（1週間）
4. ひたすら実験、試行錯誤（4週間）

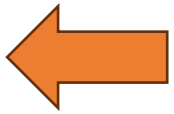
※かっこ内は今回の時間配分

2-1. 問題理解

■ 理解したいキーワード

■ 問題の背景

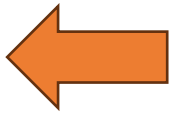
- ✓ 何を解きたいか？
- ✓ 必要そうなドメイン知識は？



- ✓ 60s 後のスペシフィックリターン予測
- ✓ Closing Cross Auction 板？

■ 評価指標（メトリクス）

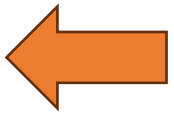
- ✓ 最適化する指標は何か？
- ✓ その指標の性質は？



- ✓ MAE のため特段考慮する必要ナシ

■ データの内容・利用可能性

- ✓ どんなデータか？
- ✓ データの流れは？（予測時点で何が利用可能か？）



- ✓ 55 timestamp / 日 ごとにその時点以前の特徴量が利用可能
- ✓ target 情報は時刻 0 時点で前日の全データにアクセス可能

2-2. ブレスト & インプット

- まずはブレストから
  - 公開情報(Notebook等)を調べる前にアイディア出しをする
  - アイディア出しの主要軸
    - ✓ 特徴量　…　結局良い特徴量が一番重要
    - ✓ 問題設計　…　多様な設計で解いたモデルのアンサンブルは大体強い

      $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$ の $\hat{\boldsymbol{y}}, f, \boldsymbol{x}$ の 3 要素それぞれを検討する

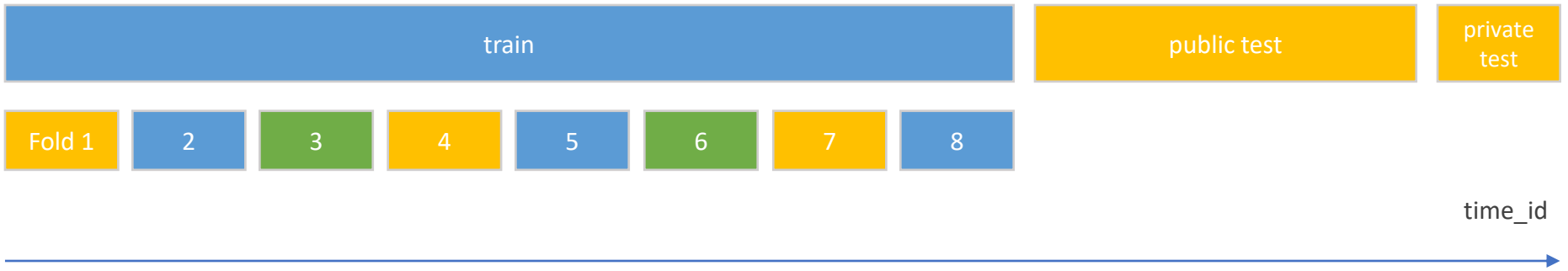
      ex. NNとGBDT両方で解いてみる（モデル f ）

      特徴量を時系列ベクトルとして入力する（入力 $\boldsymbol{x}$ ）
- 次にインプット
  - Disucussion 全部読む
  - ドメイン知識を入れる（ex. Closing Cross Auction って何？）
  - 知識を仕入れたうえで再度ブレスト

2-3. ベースライン作成

- CV（交差検証）
  - 試行錯誤の拠り所となるため、信頼できるCVの構築が重要
  - 今回は通常の KFold CV を採用（K=8）
  - メトリックは MAE とその CV 間での安定性をモニタリング

$$stable\_loss(\boldsymbol{l}) = (E[l_k]^2\sqrt{V[l_k]}\,(max(l_k) - min(l_k)))^{1/4}$$



2-3. ベースライン作成

- パイプライン実装
  - 以降の実験を高速かつ正確に回す準備をする
  - 「前処理~特徴量計算~学習~予測~後処理~評価」 をパッケージ化
  - パイプラインのテンプレートは Github で公開してます

[ ]:

```
import sys
sys.path.append('/kaggle/input/optiver-ds')
import inference

inference.run(ex_num=100, skip_no_scored=True,ignore_exception=True)
```

▲ submitは推論モジュールを呼び出すだけの図

2-4. ひたすら実験

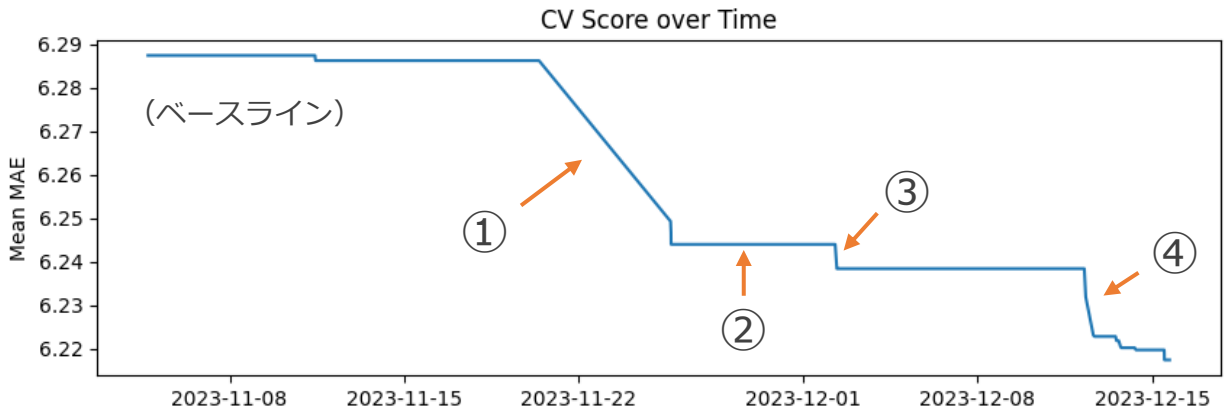
■ 検証した主な仮説

① ひたすら特徴量エンジニアリング

② 符号と絶対値の 2-stage 予測

③ Transformer

④ ブレンディング



▲ メトリック改善の時系列

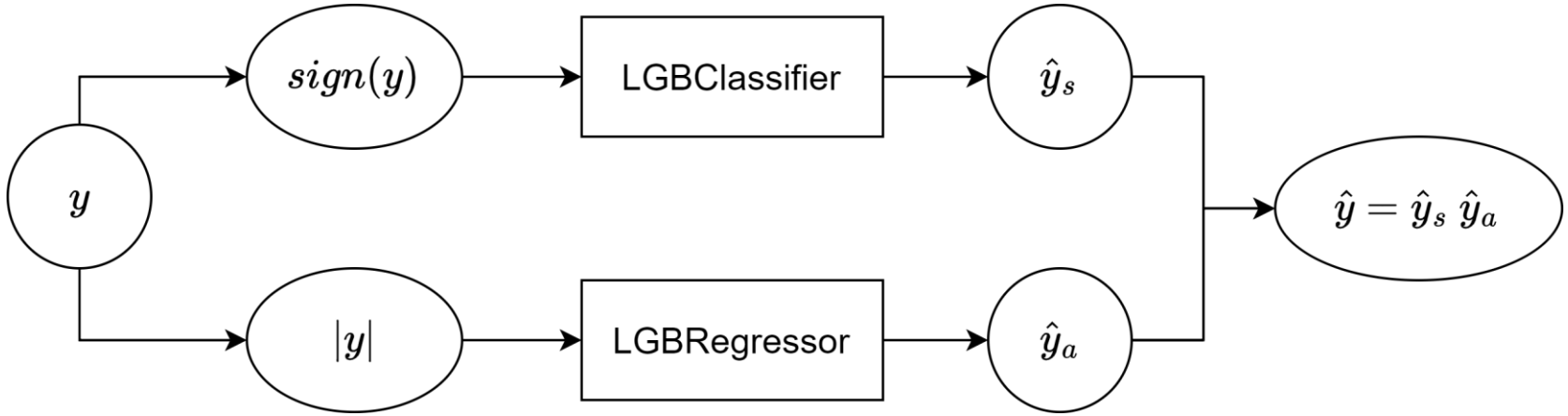
2-4-1. 特徴量エンジニアリング

- ロバストな問題設計のためか、特徴量を増やすほど精度が良くなった
- 採用した特徴量群（計 462 列）
  - 各特徴量ペアの乖離、インバランス
  - 列方向への同種の特徴量の集約
  - 前日の特徴量の集約（銘柄ごと、時間 ID ごと）
  - 銘柄軸での日内 rolling 集約特徴量
  - 各特徴量の銘柄軸、時間 ID 軸での Group Encoding
  - …etc

2-4-2.符号と絶対値の 2-stage 予測（失敗）

■ リターンの符号と絶対値を予測しうる特徴量は異なる（Prado, 2019）

→ targetを符号と絶対値に分解し、それぞれを LightGBM で学習



■ 結果：×（普通のLGBと予測傾向が似通ってしまい、アンサンブルに寄与せず）

2-4-3. Transformer

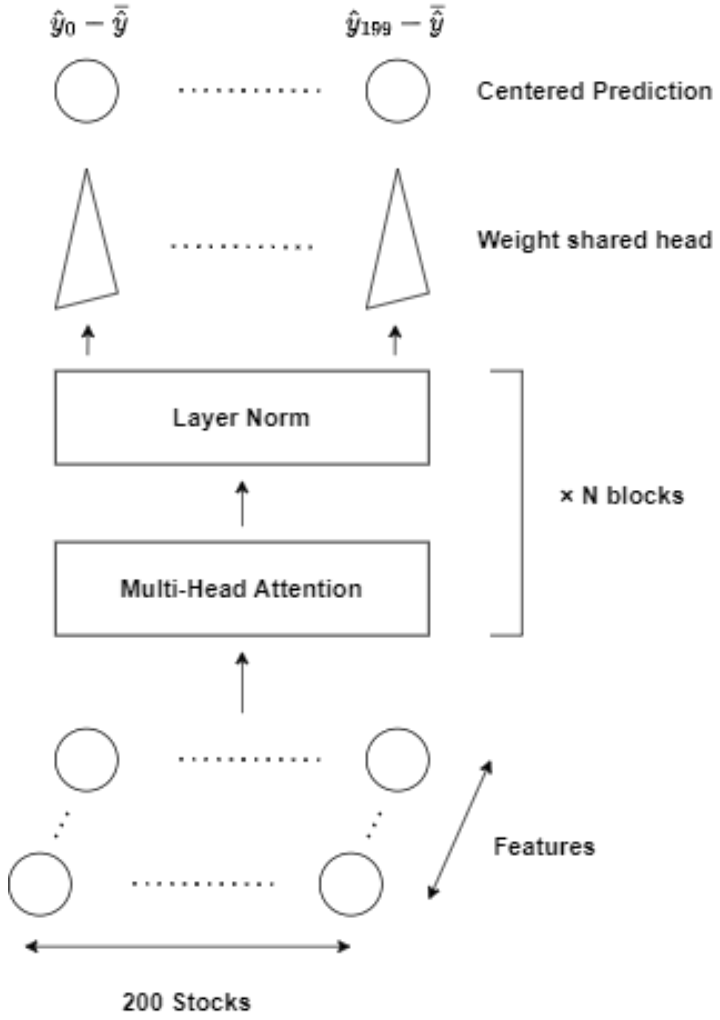
■ 目的変数はスペシフィックリターン

→他の銘柄との関連性や市況を捉えたい

→全銘柄の情報をまとめて入力 & 出力すればいいのでは？

→Attention機構なら銘柄間の関連を学習できるのでは？

⇒ Transformer Encoderいいかも？

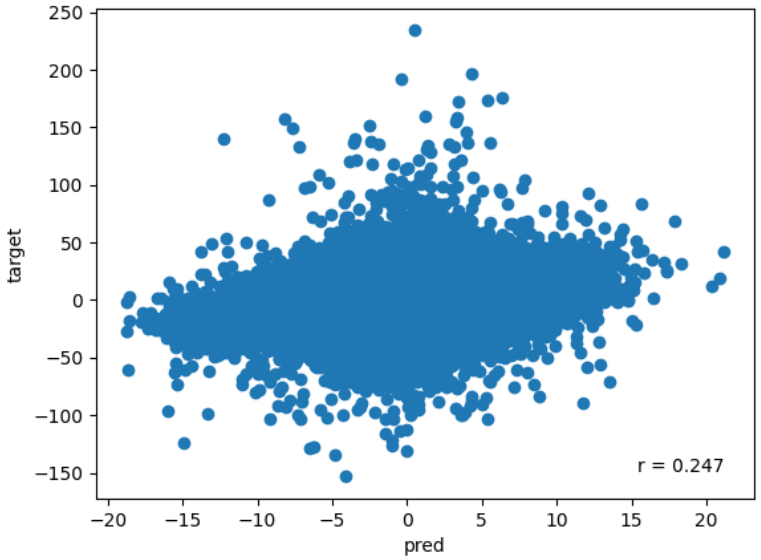


▶ モデルアーキテクチャ概要図

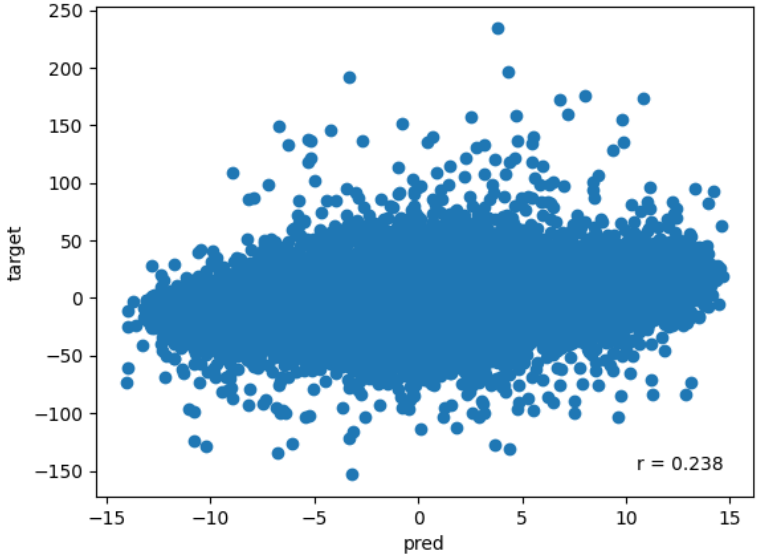
2-4-4. ブレンディング + その他

- OOF（Out of Fold）で LGB, Transformer のアンサンブル重みを最適化
- 最終的な重み … LGB：Transformer = 0.61：0.39（！）

LGB



Transformer



▲ 予測値の散布図比較。かなり傾向が異なることがわかる。

2-4-4. ブレンディング + その他

■ その他の工夫

- ✓ volume系の特徴量を敢えて前処理しない（差分化、対数化など）
- ✓ LGB のハイパラチューニングをちゃんとやる
- ✓ Transformer の学習時に mixup を適用
- ✓ 予測値をその平均で中心化

■ 効かなかった工夫

- ✓ target を Rank Gauss 変換して予測 → アンサンブル
- ✓ ボラ予測モデルを構築して予測値を補正

目次

1. ざっくり解法
2. 思考と試行の時系列振り返り
    a. 問題理解
    b. ブレスト & インプット
    c. ベースライン作成
    d. ひたすら実験
3. 上位解法との差分考察

3. 上位解法との差分考察

- （恐らく）最も大きな差分 … **「再学習」**
  - 評価期間のデータが学習データ期間とかなり離れていた
    → ドメインシフトにうまく対応できたかが重要だった
  - tonic はそもそも submit の時間制限がギリギリだった
    → NN を含めて再学習する発想に至る余裕なし
  - 1st solution は LGB + NN * 2 の構成にも拘わらず高速実装で再学習を実現
- 次点 … アンサンブルするモデル数不足
  - 時系列方向に入力をベクトル化する問題設計をうまく扱えなかった

  → CNN, GRU 等追加で検証したい

Happy Kaggling!