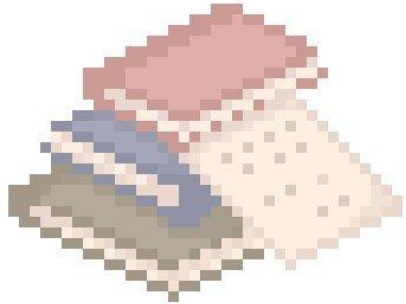
Optiver 2023 勉強会
データの基礎的な解析＋1st Solution

マケデコ勉強会
2024/05/23

自己紹介

- Yuichiro Nishimoto
  - Twitter: @nishimt general
  - Kaggle: nishimoto
  - Zenn: nishimoto



- 普段はインターネット系企業でML系エンジニア
  - 株は一般的な取引の知識程度

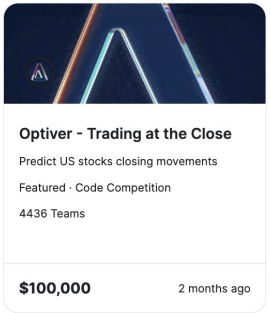
目次

- コンペ概要
- データの基礎解析
- HYDさんの解法について
  - 概要
  - Magic features（特徴量）
  - 学習モデルの比較
- 補足

目次

コンペ概要

- 板情報からNASDAQの株の価格の動きを予測するコンペ
  - MAE（平均絶対誤差）が評価指標



- データについて
  - 200銘柄×481日×55データ/日 = 523万行
  - 市場の需給、価格、量に関する特徴量がある（17列）

コンペ概要

- 評価データの期間は以下の通り

https://www.kaggle.com/competitions/optiver-trading-at-the-close/discussion/459329#2548844

目次

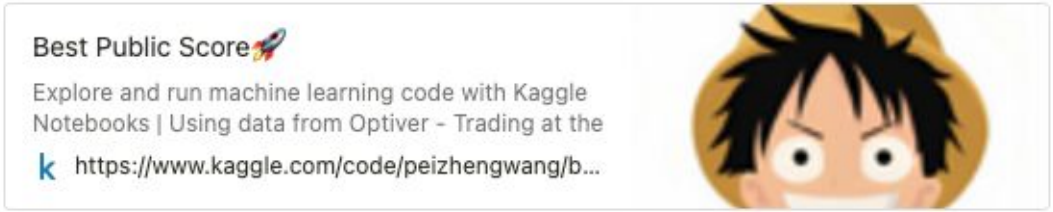
- コンペ概要
- **データの基礎解析**
- HYDさんの解法について
  - 概要
  - Magic features（特徴量）
  - 学習モデルの比較
- 補足

データの基礎解析 - 目次

1. 検証で使用したベースラインコードについての紹介
2. データについての基礎的な解析
3. ベースラインコード特徴量とTarget値の相関
4. オンライン学習の必要性

1. 検証で使用したベースラインコードについて

- ALEX WANGさんのコードを参考にしていた
  - https://www.kaggle.com/code/peizhengwang/best-public-score
  - 160特徴量
    - 各売買/需給の価格や量のバランス
    - 各売買/需給の価格や量の過去値や移動平均値
  - Online learning（追加データでの学習）はなし
  - 学習モデルはLightGBM



今日の検証コードは全部Kaggleにて公開しています。補足参照

2. データについての基礎的な解析

- データについて
  - 需給や価格に関する特徴量がある

需給系の特徴量	matched/imbalance_size（需給マッチ/アンマッチ量） imbalance_buy_sell_flag（需給マッチのフラグ）
価格系の特徴量	bid/ask_price, refrence_price, wap（売・買の提示価格，参照価格，WAP（加重平均価格） bid/ask_size（売・買の量） far/near_price（買値と売値がクロスした価格；far priceは成行を除外）

- 欠損値
  - far/near priceは半数くらいが欠損値
  - その他列もわずかにNaNがある（0.01%程度）

2. データについての基礎的な解析

- Targetは以下のように計算

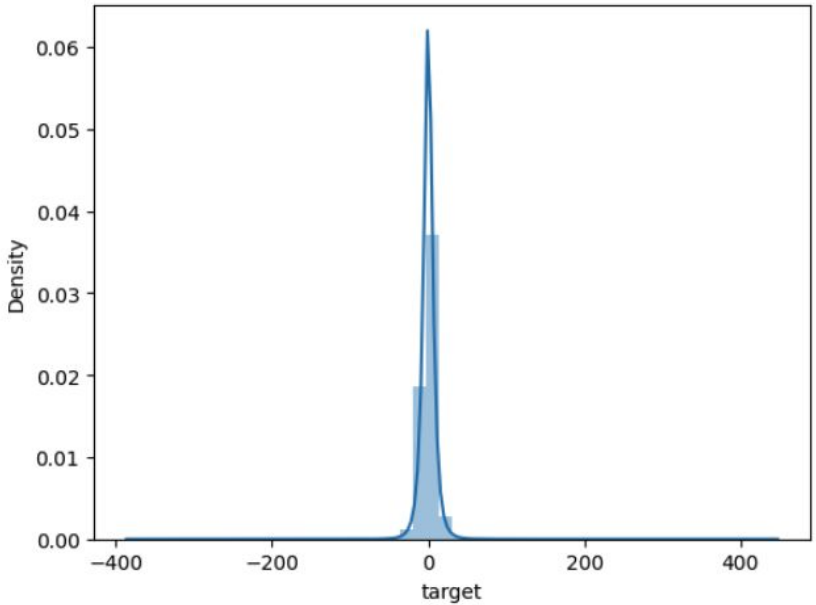
1. wapの1分前の値との変化倍率から「各株のリターン」を算出
   （wapなどの価格系は正規化済み）

2. stock_idごとの重みを元に「重み付け平均リターン」を算出
   （重み付けはOptiverが時価総額を元に?決めてる）

3. 「各株のリターン」から「重み付け平均リターン」を引き算してtargetを算出

2. データについての基礎的な解析 – target値の分布

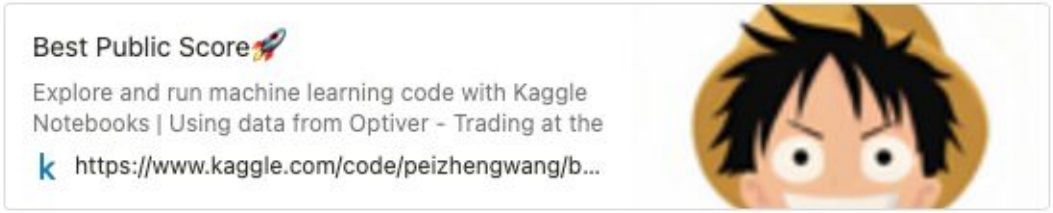
- 値の分布をみたところ、裾が広すぎる。。



平均値	-0.0476
標準偏差	9.4529
最小値	-385.2898
25%値	-4.5598
中央値	-0.0602
75%値	4.4096
最大値	446.0704

3. ベースラインコード特徴量とTarget値の相関

- ALEX WANGさんのNotebookを元に特徴量を作成



- 全期間での相関係数を算出
  - Kaggle Notebookに詳細なコード有

3. ベースラインコード特徴量とTarget値の相関

・ピアソン/スピアマン相関係数の高い/低いもの5つを表示

・最大は0.10を超えるなどかなり相関が高い

	Pearson	Spearman	Mean
reference_price_wap_imb	0.112435	0.107523	0.109979
ask_price_wap_imb	0.089753	0.095778	0.092766
bid_price_wap_imb	0.087378	0.095712	0.091545
ask_price_bid_price_wap_imb2	-0.000554	0.129163	0.064305
ask_size	0.014936	0.067094	0.041015
...	...	...	...
wap_momentum	-0.040366	-0.044928	-0.042647
bid_size	-0.018000	-0.068349	-0.043174
size_imbalance	-0.022664	-0.129272	-0.075968
liquidity_imbalance	-0.114617	-0.129272	-0.121944
market_urgency	-0.144134	-0.139411	-0.141773

x_y_imb = (x − y) / (x+y)

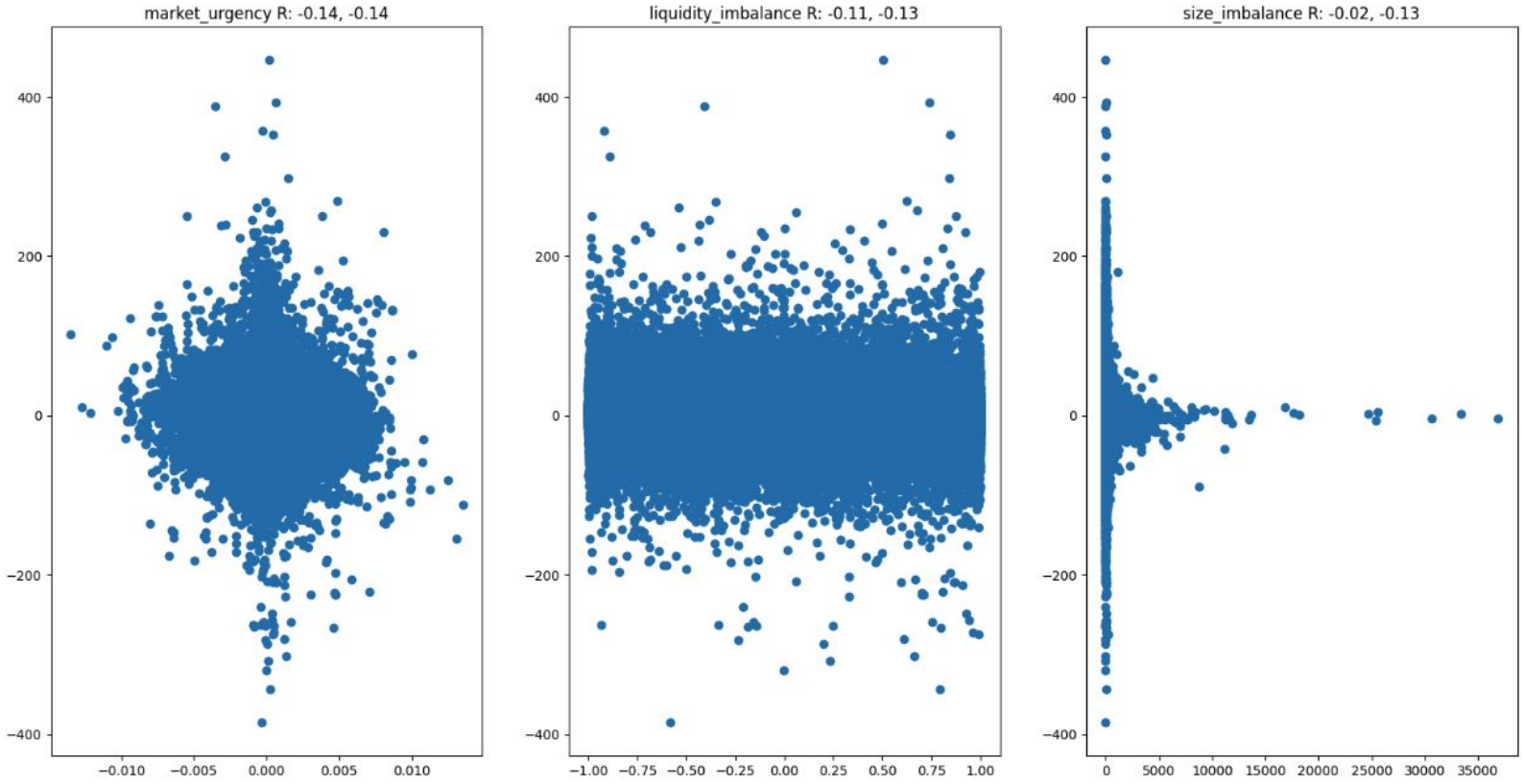
size_imbalance = bid_size / ask_size

liquidity_imbalance = (bid_size − ask_size) / (bid_size + ask_size)

market_urgency = (ask_price − bid_price) * liquidity_imbalance

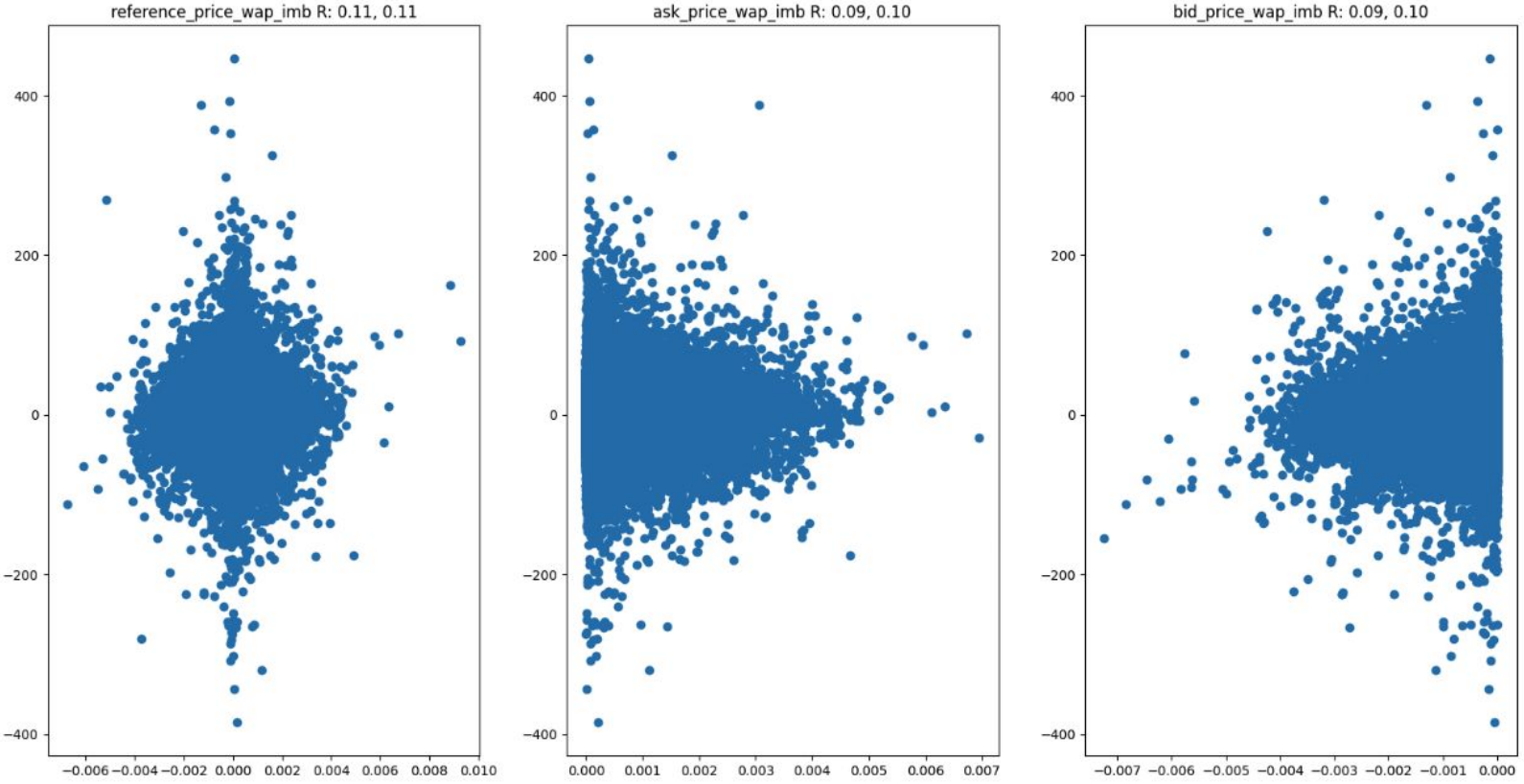
3. ベースラインコード特徴量とTarget値の相関

散布図は以下の通り（xが各指標, yがtarget）



3. ベースラインコード特徴量とTarget値の相関

散布図は以下の通り（xが各指標, yがtarget）



4. オンライン学習の必要性

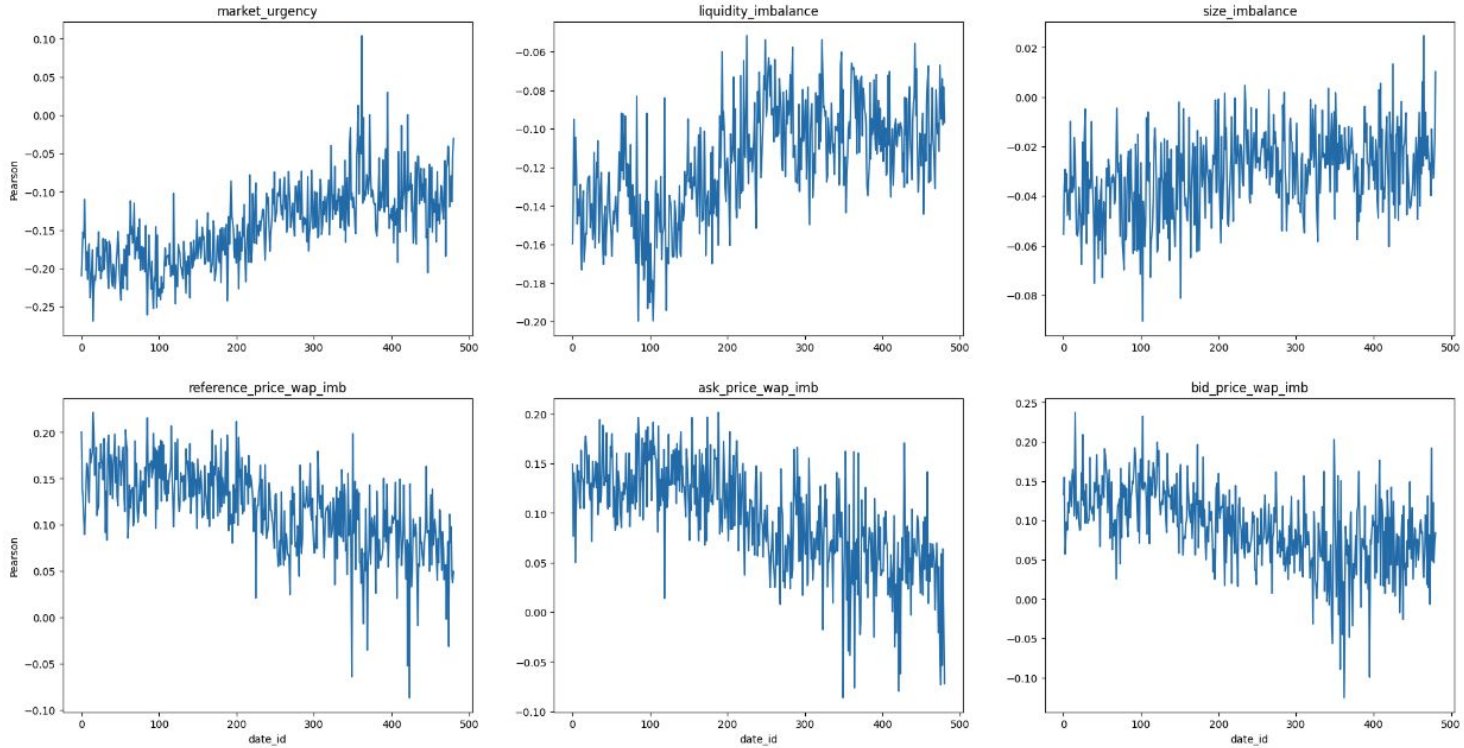
・今回、コンペ上位の方の多くがオンライン学習（予測用に配布された新たなデータから再学習すること）を採用していた

→ なお、再学習の期間は上位陣でもバラバラ

・どうやったらこれに気がつけたか？を考えてみた

4. オンライン学習の必要性

先ほど、相関が高かった変数の相関係数をdate_idごとにplot



元々相関が高い変数は多くが特徴が弱くなっている → オンライン学習必要?

目次

- コンペ概要
- データの基礎解析
- **HYDさんの解法について**
  - **概要**
  - **Magic features（特徴量）**
  - **学習モデルの比較**
- 補足

HYDさんの解法概要

- URL: https://www.kaggle.com/competitions/optiver-trading-at-the-close/discussion/487446
- 300特徴量をCatboost, GRU, Transformerでアンサンブル
  - 重さは0.5, 0.3, 0.2なので、Catboostが強め

- オンライン学習と後処理がスコアアップにつながった
  - オンライン学習は12日ごと(5回)に追加学習
  - 今回は**特徴量、各種モデル**(Catboost、GRU、Transformer)について深堀りします

model name	validation set w/o PP	validation set w/ PP	test set w/o OL w/ PP	test set w/ OL one time w/ PP	test set w/ OL five times w/ PP
CatBoost	5.8287	5.8240	5.4523	5.4291	5.4165
GRU	5.8519	5.8481	5.4690	5.4368	5.4259
Transformer	5.8614	5.8619	5.4678	5.4493	5.4296
GRU + Transformer	5.8233	5.8220	5.4550	5.4252	5.4109
CatBoost + GRU + Transformer	5.8142	5.8117	5.4438	5.4157	5.4030*(overtime)

目次

- コンペ概要
- データの基礎解析
- HYDさんの解法について
  - 概要
  - **Magic features（特徴量）**
  - 学習モデルの比較
- 補足

Magic features（特徴量）

Magic Features

My models have 300 features in the end. Most of these are commonly used, such like raw price, mid price, imbalance features, rolling features and historical target features.

I will introduce some features really helpful and other teams didn't share yet.

1 agg features based on seconds_in_bucket_group

```
pl.when(pl.col('seconds_in_bucket') < 300).then(0).when(pl.col('seconds_in_bucket') <
480).then(1).otherwise(2).cast(pl.Float32).alias('seconds_in_bucket_group'),
```

```
 *[(pl.col(col).first() / pl.col(col)).over(['date_id', 'seconds_in_bucket_group',
'stock_id']).cast(pl.Float32).alias('{}_group_first_ratio'.format(col)) for col in base_features],
 *[(pl.col(col).rolling_mean(100, min_periods=1) / pl.col(col)).over(['date_id', 'seconds_in_bucket_group',
'stock_id']).cast(pl.Float32).alias('{}_group_expanding_mean{}'.format(col, 100)) for col in base_features]
```

2 rank features grouped by seconds_in_bucket

```
 *[(pl.col(col).mean() / pl.col(col)).over(['date_id',
'seconds_in_bucket']).cast(pl.Float32).alias('{}_seconds_in_bucket_group_mean_ratio'.format(col)) for col in
base_features],
 *[(pl.col(col).rank(descending=True,method='ordinal') / pl.col(col).count()).over(['date_id',
'seconds_in_bucket']).cast(pl.Float32).alias('{}_seconds_in_bucket_group_rank'.format(col)) for col in
base_features],
```

Magic features（特徴量）

いくつかの条件でグループ化してその比を特徴量として足している。具体的には以下

・同一銘柄/秒群間の比率特徴

- 銘柄/日/秒群でグループ → 最初の秒数の値との比率
- 銘柄/日/秒群でグループ → 移動平均値との比率

・同一秒/銘柄間の比率特徴

- 日/秒でグループ → 平均値との比率
- 日/秒でグループ → 値のランクの百分位数

Magic features（特徴量）

①銘柄/日/秒群でグループ → 最初の値との比率

②銘柄/日/秒群でグループ → 移動平均値との比率

Price値を集計

【値の例（同日・別秒・同銘柄）】

date_id	seconds_in_bucket	stock_id	price
1	0	2	100
1	10	2	200
1	20	2	300
1	310	2	400
1	320	2	500
1	330	2	600
1	510	2	700
1	520	2	800
1	530	2	900

Magic features（特徴量）

①銘柄/日/秒群でグループ → 最初の値との比率

②銘柄/日/秒群でグループ → 移動平均値との比率

秒で群分け
0-290: 0
300-470: 1
480-540: 2

群の最初の値との比
100 / 100 = 1
100 / 200 = 0.5
....

移動平均(Max 100)との比
Avg(100) / 100 = 1
Avg(100, 200) / 200 = 0.75
Avg(100, 200, 300) / 300 = 0.67

【値の例（同日・別秒・同銘柄）】

date_id	seconds_in_bucket	stock_id	price	seconds_in_bucket_group	price_group_first_ratio	price_group_expanding_mean100
1	0	2	100	0	1.000	1.000
1	10	2	200	0	0.500	0.750
1	20	2	300	0	0.333	0.667
1	310	2	400	1	1.000	1.000
1	320	2	500	1	0.800	0.900
1	330	2	600	1	0.667	0.833
1	510	2	700	2	1.000	1.000
1	520	2	800	2	0.875	0.938
1	530	2	900	2	0.778	0.889

Magic features（特徴量）

③日/秒でグループ → 平均値との比率

④日/秒でグループ → 値のランクの百分位数

【値の例（同日・**同秒**・**別銘柄**）】

date_id	seconds_in_bucket	stock_id	price
1	0	2	100
1	0	3	200
1	0	4	300
1	0	5	400
1	0	6	500
1	0	7	600
1	0	8	700
1	0	9	800

Magic features（特徴量）

③日/秒でグループ → 平均値との比率

④日/秒でグループ → 値のランクの百分位数

同一秒の平均値との比率
450 / 100 = 4.5
450 / 200 = 2.3
…

同一秒の値のランクを割合化
8/8 = 1.000
7/8 = 0.875
…

【値の例（同日・**同秒**・**別銘柄**）】

date_id	seconds_in_bucket	stock_id	price	price_seconds_in_bucket_group_mean_ratio	price_seconds_in_bucket_group_rank
1	0	2	100	4.500	1.000
1	0	3	200	2.250	0.875
1	0	4	300	1.500	0.750
1	0	5	400	1.125	0.625
1	0	6	500	0.900	0.500
1	0	7	600	0.750	0.375
1	0	8	700	0.643	0.250
1	0	9	800	0.563	0.125

Magic features（特徴量）– 検証

・Solution内ではどの特徴量に対して集計を行ったか明記されていない

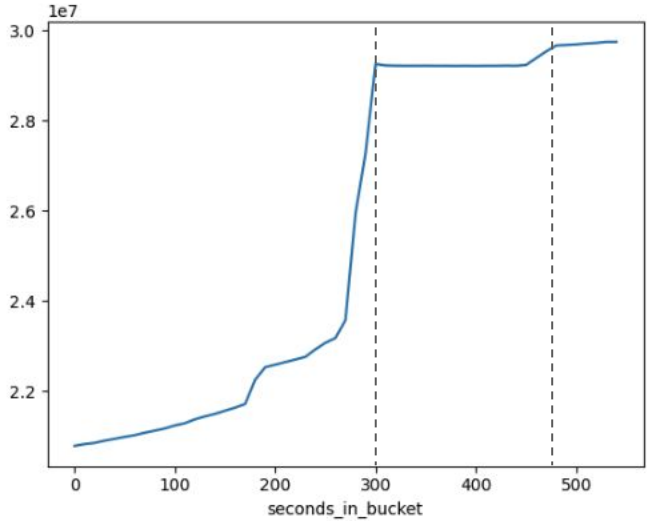
→ ask/bid priceとask/bid sizeについて集計特徴量を追加したところ、評価指標が**CVとPB共に改善**

（CV：5.860 => 5.858；PB：5.480 => 5.478）

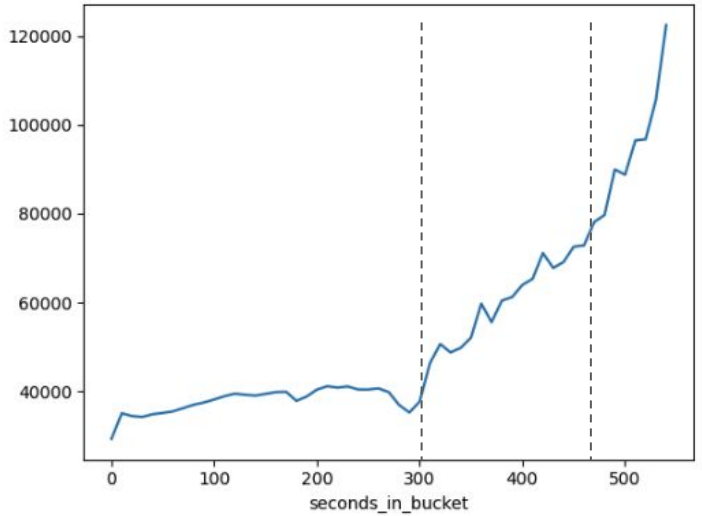
Magic features（特徴量）– なぜ改善するか？

・各秒群内での特徴や、同秒内の各銘柄の相対的な特徴であると考えられる
→ 300秒と480秒で区切った理由も語られていないが、各種sizeを見てみると確かに挙動が違うためこれら秒数を設定したのかも

matchedとunmatched sizeの合計の平均値



bidとask sizeの合計の平均値



目次

- コンペ概要
- データの基礎解析
- HYDさんの解法について
  - 概要
  - Magic features（特徴量）
  - **学習モデルの比較**
- 補足

学習モデルの比較

・hydさんはCatboost, GRU, Transformerを使用していた

・LightGBMと各種モデルを比較した

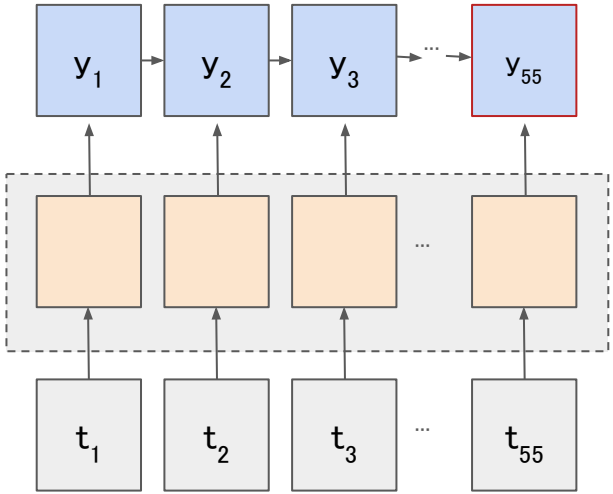
	特徴（やLightGBMと比較したときの強み/弱み）
Catboost	・LightGBMと同様の勾配ブースティング系のツール 👍 カテゴリカル変数に強い 👍 精度がよい（こともある） 👎 メモリ使用量が多い、学習や推論が遅い
GRU	・時系列予測でたまに使われる （自分の実装ではメモリが不足したので検証対象外）
Transformer	・系列予測でたまに使われる（ KaggleだとRNAの反応値予測にTransoformerが使用されていた） 👍 他の系列の特徴量も加味した予測を行ってくれる 👎 メモリ使用量が多い

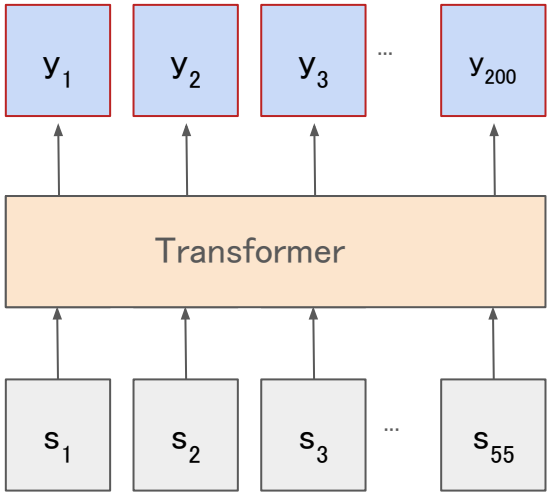
学習モデルの比較 – GRUとTransformerについて補足

- GRUは各銘柄の55タイムポイントを同時に入力に使用 => 最終タイムポイントだけ予測に利用
- Transformerは同タイムポイントの200銘柄を同時に入力に使用 => 200銘柄の出力を同時予測

GRUに時系列情報の学習、Transformerに銘柄間の情報を学習させることが目的だったとのこと(参考)

GRU





理論的な説明はアイシアさんがわかりやすいです(GRU、Transformer)

学習モデルの比較（精度比較）

	CV（後半2割データ）	Private
LightGBM	**5.858**	**5.478**
Catboost	5.865	5.483
Transformer	5.923	5.613
LightGBM+Transofrmer （8:2でアンサンブル）	-	5.482

自分の実装ではLightGBMに勝てるモデルは作れなかった。。
以下、使用感について軽くまとめています

使ってみた感想（1）Catboost

- Catboostは後述のニューラルネットと比較すると使いやすかったが、LightGBMと比較すると精度は落ちていた

- 今回自分がカテゴリカル変数に設定したものは以下
  - stock_id（銘柄番号）
  - dow（曜日）
  - seconds（秒）
  - minute（分）
  - imbalance_buy_sell_flag（需給のアンマッチが売買どちら側で起きているか）
  - imbalance_buy_sell_flag_shift_1/3/5/10（上記変数の1/3/5/10行前の状態）

- 予測に有効なカテゴリカル変数があれば精度改善する（かも）

使ってみた感想(2)ニューラルネット系モデル

- GRU/Transformerは検討することが多い/エンジニアリング的にも大変
- 今回具体的に試行錯誤したものは以下

検討するもの	今回の対応
正規化	z-score化を採用した
NaN値の扱い	全体平均値で埋めた その日の平均値で埋めるなどいろいろ工夫できそう
ハイパーパラメータ	GRUのhidden size、Transformerいくつ重ねるか、など 勾配ブースティングより経験が少なく試行錯誤の手間は大きい

- 勾配ブースティングはNaN値の扱いに長けている印象なので、今回のコンペではあまりNNが使われなかった(かも)

使ってみた感想(2)ニューラルネット系モデル

- GRUは予測する秒によって入力が異なる点も気になった
  - 例
  - x日の0秒目はx-1日の10秒～x日の0秒が入力
  - x日の550秒目はx日の0-550秒が入力

- エンジニアリング的な難しさも増す
  - 実際回したところ、GRUはメモリ不足で詰まり、Transformerは実行時間で苦労した。
  - TransformerがLGBの5-7倍くらいの実行時間
  - 実運用でも予測が間に合わないなどの可能性もある

Thank you!

補足：コード

全コードKaggleにおいてあります

基礎解析（EDA）	https://www.kaggle.com/code/nishimoto/opt23-makedeco-eda
LightGBM	https://www.kaggle.com/code/nishimoto/opt23-makedeco-lgb-add-data
Catboost	https://www.kaggle.com/code/nishimoto/opt23-makedeco-cat-add-data
Transformer（学習）	https://www.kaggle.com/code/nishimoto/opt23-makedeco-transformer?scriptVersionId=175668233
Transformer（推論）	https://www.kaggle.com/code/nishimoto/opt23-makedeco-transformer?scriptVersionId=177204998