CS5489 Course project

Xian Jia Le,Ben
56214537
Group : 58
optiver-trading at the close

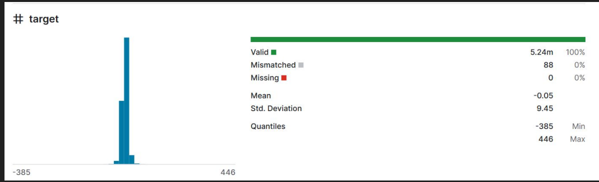
Intro

1. Given 17 features
2. Predict the target price
3. Regression problem

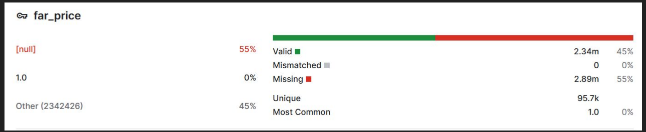
# imbalance_size	# imbalance_bu...	# reference_pri...	# matched_size	far_price	near_price	# bid_price
0.0	0	1.000635	13552875.92			0.999779
969969.4	1	1.000115	3647503.98			0.999506
9412959.1	1	0.999818	21261245.87			0.999741
2394875.85	1	0.999916	9473209.08			0.999022
3039700.65	-1	1.000969	6248958.45			0.999354
10482752.19	-1	1.001374	8839457.1			0.999885
1506120.2	-1	0.999968	2001112.44			0.99984
11739945.44	1	0.999794	13597118.7			0.999794
5749286.01	1	0.998995	7039173.61			0.999074

Data Cleaning

1. Drop record without valid target



2. Fill NAN as as 0



3. Reduce memory by changing data type
    a. allow training multiple models at the same time

Feature Engineering

1. Random combination of attributes (Up to 5.3660)

```
prices = ["reference_price", "far_price", "near_price", "ask_price", "bid_price", "wap"]
for c in combinations(prices, 2):
    #df[f'{c[0]}_-_{c[1]}'] = (df[f'{c[0]}'] - df[f'{c[1]}']).astype(np.float32) ## difference between the different prices
    df[f"{c[0]}_{c[1]}_imb"] = df.eval(f"({c[0]} - {c[1]})/({c[0]} + {c[1]})")
```

2. Time related
   a. Relative data to previous day

```
for col in ['matched_size', 'imbalance_size', 'reference_price', 'imbalance_buy_sell_flag']:
    for window in [1, 2, 3,10]:
        df[f"{col}_shift_{window}"] = df.groupby('stock_id')[col].shift(window) # keep track for previous value
        df[f"{col}_ret_{window}"] = df.groupby('stock_id')[col].pct_change(window) # percentage change with previous value
```

   b. Statistical Data

```
global_stock_id_feats = {
    "median_size": df.groupby("stock_id")["bid_size"].median() + df.groupby("stock_id")["ask_size"].median(),
    "std_size": df.groupby("stock_id")["bid_size"].std() + df.groupby("stock_id")["ask_size"].std(),
    "ptp_size": df.groupby("stock_id")["bid_size"].max() - df.groupby("stock_id")["bid_size"].min(),
    "median_price": df.groupby("stock_id")["bid_price"].median() + df.groupby("stock_id")["ask_price"].median(),
    "std_price": df.groupby("stock_id")["bid_price"].std() + df.groupby("stock_id")["ask_price"].std(),
    "ptp_price": df.groupby("stock_id")["bid_price"].max() - df.groupby("stock_id")["ask_price"].min(),
}
```

Training Strategy

1. Model :LGBM (accuracy : CATBoost >= LGBM > XGB)
   a. relatively fast and accurate
2. Combine multiple models (5 Models)
3. Each of them trains with different amount of data

Strategies:

1. Each model trained with different equal amount sub-data
2. Model is trained with increase amount of sub-data (start at 288 day, 48 step)
   a. Start amount from the half of data

Predict Strategy

1. Accumulate the data from previous loop
2.
```
cache = pd.concat([cache, test], ignore_index=True, axis=0)
if counter > 0: # take 4 week as windows
    cache = cache.groupby(['stock_id']).tail(28).sort_values(by=['date_id', 'seconds_in_bucket', 'stock_id']).reset_index(drop=True)
```
3. Predict the result from models
4. Average the result

Final Result:

5.3714 (10 LGBMs)

5.3608 (5 LGBMs)

5.3657 (1 LGBM)

Limitations

1. Sensitive to training strategy
2. Time Comsuming -> Early stop is required

Things can be tried

1. Adding rolling features
   a. takes average with previous 5 days

2. Eliminate less important features

3. Ensemble different type of models

4. Create features which are formed by knowledge from domain area (Finance)

Covariance between target and attributes

target	1.000000
liquidity_imbalance	-0.114617
reference_price_wap_imb	0.112434
reference_price_-_wap	0.112425
ask_price_wap_imb	0.089749
	...
far_price_/_near_price	-0.000076
far_price_/_bid_price	-0.000027
reference_price_bid_price_wap_imb2	0.000025
far_price_/_ask_price	-0.000025
far_price_/_wap	-0.000012

Thank you