

QUANT RUSH · PICKS & SHOVELS · WORKSHOP 3

REFINING THE ORE

From raw columns to features and first models.

TUE NOV 3

FEATURES + BASELINES

Engraving: H. Sandham, “The Sluice,” The Century, 1883 · Public domain

TODAY

From raw columns to first models

1

BUILD

Ten features from the raw columns, each with an idea behind it.

2

MEASURE

Rank them within the era, then check what each one is worth.

3

FIT

A linear model and gradient-boosted trees, checked honestly.

Every number here is from the training data, and the notebook reproduces it.

A feature is a column with an idea

A model only sees the columns you give it. A linear model can add columns up, but it can’t compute a spread, a maximum, or a correlation over 12 hours by itself.

A feature hands it the idea directly, as one new column:

```
ret_sum_4 = ret_00 + ret_01 + ret_02 + ret_03
```

Today: ten features like this one, each from a single row.

TWO RULES

Use one row or one era: the asset’s own 12 hours, or the other assets at the same moment.

Never use the target, not even through an average of it.

TEN FEATURES, FOUR IDEAS

Built from each row's own 12 hours

REVERSAL OVER HOURS

`ret_sum_2 · ret_sum_4 · ret_sum_12`

The sum of the latest 2, 4 and 12 hourly returns.

JUMPINESS

`ret_std · ret_maxabs`

The spread of the 12 returns, and the largest move either way.

RECENT ACTIVITY

`vol_recent · range_recent · taker_recent`

The average of the latest 4 hours. vol and range rows average 0, so this is "recent minus usual."

MOVES WITH THE MARKET

`mkt_beta · mkt_corr`

Beta and correlation of the rebuilt raw returns, mkt_ret + ret × mkt_disp, with the market's.

RANK WITHIN THE ERA

One scale in every era

mkt_beta on the last 1,080 training eras:

	As computed	Ranked
lowest	-5.584	-0.495
median	1.017	0.003
highest	7.149	0.500

```
f.groupby(era).rank(pct=True) - 0.5
```

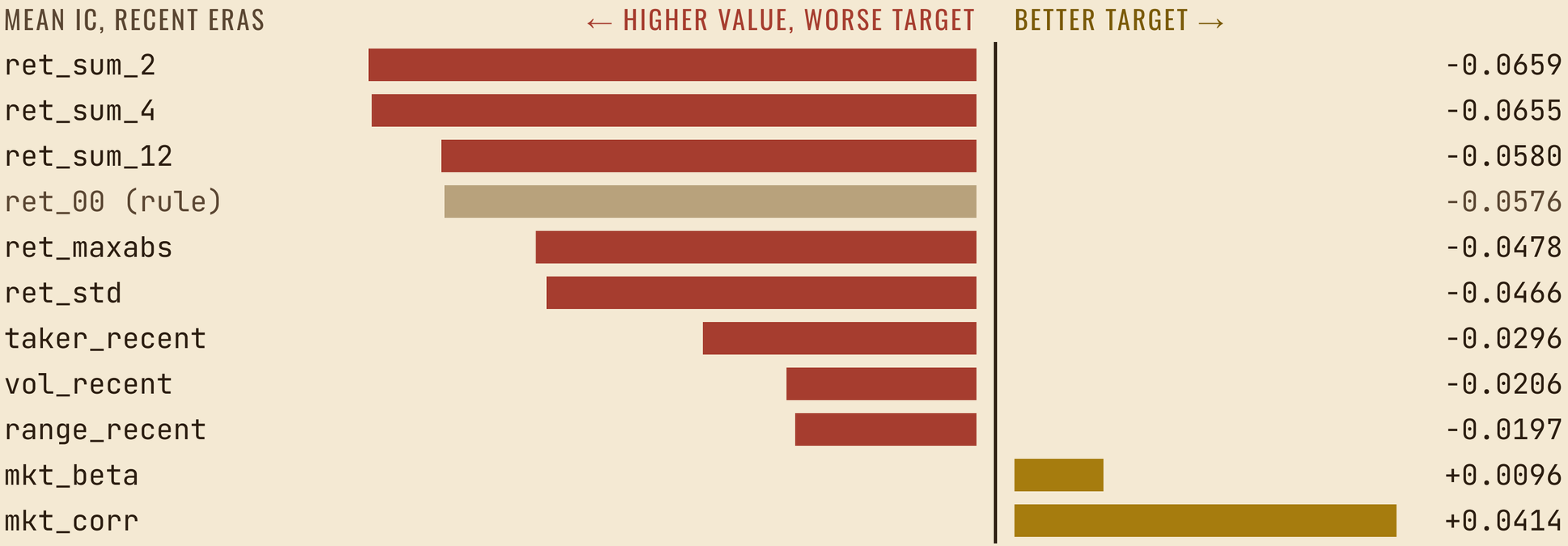
WHY RANK

The target is a rank within the era, and the metric only compares assets within an era.

Ranking keeps that order, so the IC doesn't change. But a model now sees one scale, with no outliers.

MEASURE EACH FEATURE

Several hours beat one



IS IT REALLY NEW?

Check what a feature overlaps with

Correlation of within-era ranks, recent eras:

Pair	Corr.	Reading
ret_std, ret_maxabs	0.95	Near-copies
mkt_corr, ret_std	−0.77	Mostly jumpiness
ret_sum_2, ret_sum_4	0.63	Related, not equal

WHAT RIDGE MAKES OF IT

Its weights, next to the 72 inputs:

```
ret_std      -0.077
ret_maxabs   +0.016
mkt_corr     -0.002
```

Given ret_std, the other two add almost nothing.

Overlap isn't harmful, but it adds less than the IC suggests.

THE CLASSIC BASELINES

A weighted sum, or many small trees

`Ridge`

LINEAR MODEL

A weighted sum of the columns. Fast and hard to overfit, but it sees only straight-line effects, one column at a time.

`HistGradientBoostingRegressor`

GRADIENT-BOOSTED TREES

Many small trees, each correcting the last. They find curves and combinations by themselves. LightGBM is the faster twin.

HOW EVERY MODEL IS CHECKED (TRAIN ERAS, OLDEST TO NEWEST)



RESULTS

Every model beats the rule

Scored with the competition metric on the last 1,080 training eras:

Model	Inputs	Score
Rule: -ret_00	1 column	0.0576
Ridge	72 inputs	0.0761
Ridge	72 inputs + 10 features	0.0888
Gradient-boosted trees	72 inputs	0.0921
Gradient-boosted trees	72 inputs + 10 features	0.0907

A model combines many weak signals; one column can't.

THE LESSON

Ridge needs your features. Trees build their own.

Features lift Ridge from 0.0761 to 0.0888. The trees go from 0.0921 to 0.0907, and a new random seed alone moves them by about as much.

A gap that small is not evidence either way. Telling luck from skill is Workshop 4.

IN CODE

Rank, stack, fit, score

```
def rank_in_era(df, era):
    return df.groupby(era).rank(pct=True) - 0.5

def make_X(df):
    feats = rank_in_era(make_features(df), df["era"])
    return pd.concat([df[INPUTS], feats], axis=1)

model = Ridge(alpha=1.0).fit(make_X(early), early["target"])
recent["prediction"] = model.predict(make_X(recent))
print(round(era_score(recent), 4))  # 0.0888
```

make_features, early and recent are in the notebook. Swap in the trees and nothing else changes.

PITFALLS

Four ways to fool yourself

1 TARGET LEAKAGE

No feature built from the target, or from an average of it. The test set has no target.

2 LINKING ERAS

No features that link rows across eras or use the test set's layout. It's against the rules.

3 SCORING ON SEEN ERAS

Fit on earlier eras, leave a gap, score on the latest. Never score where the model learned.

4 TUNING TO THE CHECK

Every setting you try on the recent eras is a small peek at them. Small gains may be luck.

Red: against the rules. Brown: legal, but it misleads you.

TRY THIS WEEK

Four things to try in the notebook

1 ADD A FEATURE

Scale ret_sum_4 by ret_std, as the target is scaled by volatility. With Ridge: 0.0888 to 0.0891.

2 REMOVE FEATURES

Does Ridge need all ten? Does it need the 72 inputs at all, next to the features?

3 BLEND TWO MODELS

Rank Ridge with features and the trees within each era, then average. We got 0.0927.

4 SUBMIT

Refit your pick on all of train, rank within each era, run the checks, upload.

Gains of a thousandth or two may be luck: keep the ideas, not the decimals.

RECAP

Ore refined, first models fit

1. A feature is a column with an idea, from one row or one era.
2. Rank it within the era, measure its IC, check its overlap.
3. Ridge needs features; trees build many of their own.
4. Fit early, score recent, then refit on all of train to submit.

Questions are welcome on Discord. Dig well.

Next: Tue Nov 10

WHAT GLITTERS

Workshop 4: validation and overfitting. When is a better score real?