# Dataset Description

The competition dataset comprises a set of timeseries with 79 features and 9 responders, anonymized but representing real market data. The goal of the competition is to forecast one of these responders, i.e., `responder_6`, for up to six months in the future.

You must submit to this competition using the provided Python evaluation API, which serves test set data one timestep by timestep. To use the API, follow the example in this notebook. (Note that this API is different from our legacy timeseries API used in past forecasting competitions.)

```
import pandas as pd
import numpy as np
import polars as pl
from matplotlib import pyplot as plt
from matplotlib.ticker import MaxNLocator, FormatStrFormatter,
PercentFormatter
import seaborn as sns

ROOT_DIR = "/kaggle/input/jane-street-real-time-market-data-
forecasting"
```

# Features

- features.csv - metadata pertaining to the anonymized features

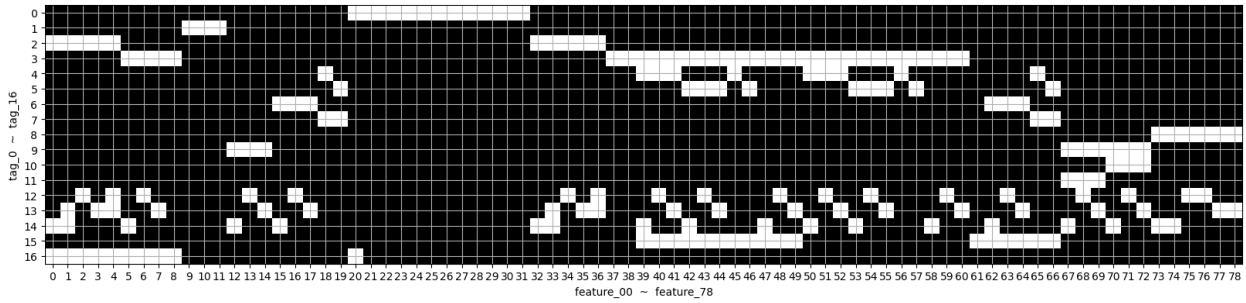
```
features = pd.read_csv(f"{ROOT_DIR}/features.csv")
features
```

```
       feature  tag_0  tag_1  tag_2  tag_3  tag_4  tag_5  tag_6  tag_7
tag_8  \
0   feature_00  False  False   True  False  False  False  False  False
False
1   feature_01  False  False   True  False  False  False  False  False
False
2   feature_02  False  False   True  False  False  False  False  False
False
3   feature_03  False  False   True  False  False  False  False  False
False
4   feature_04  False  False   True  False  False  False  False  False
False
..         ...    ...    ...    ...    ...    ...    ...    ...    ...
...
74  feature_74  False  False  False  False  False  False  False  False
True
75  feature_75  False  False  False  False  False  False  False  False
True
```

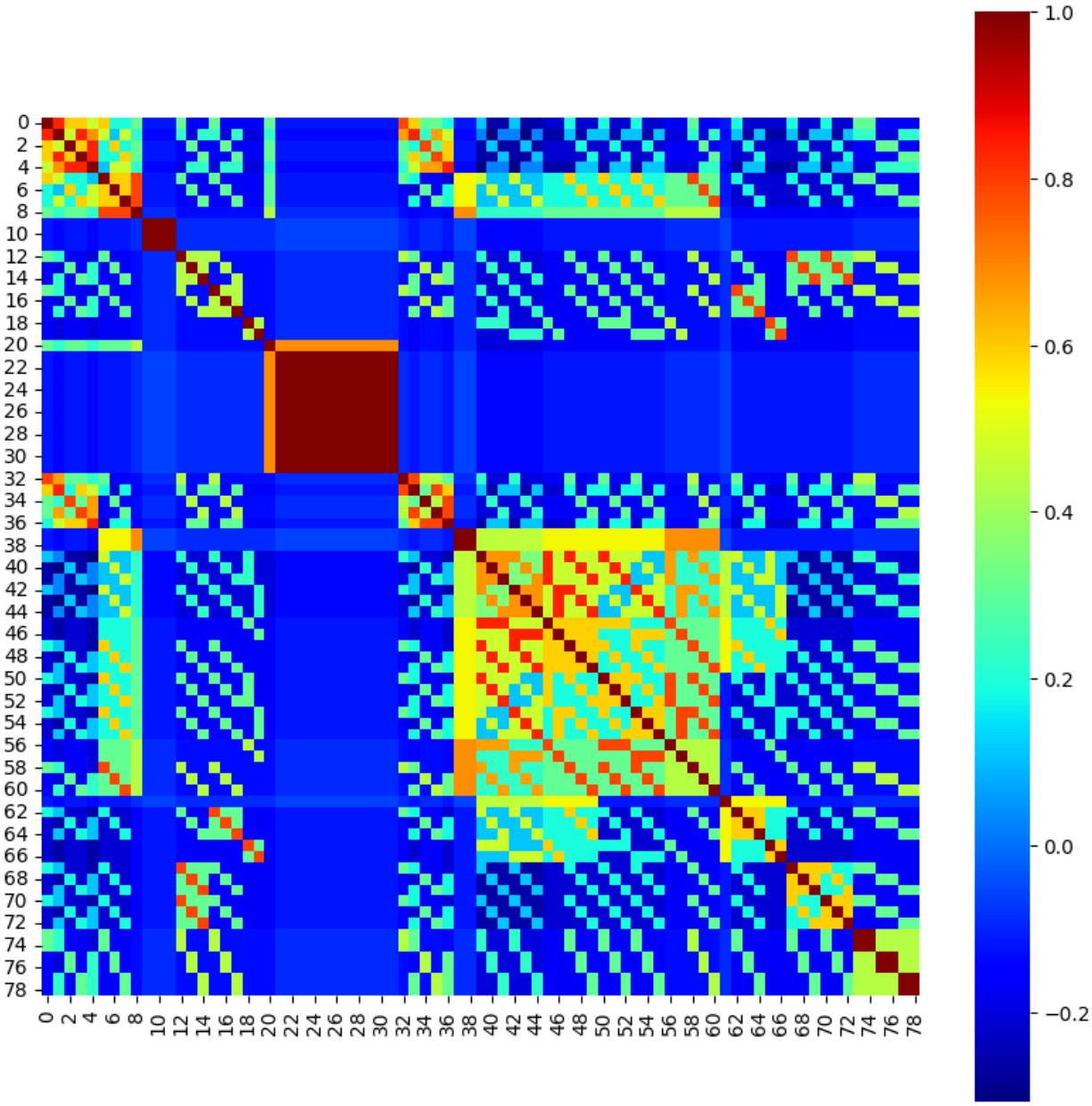
```
76  feature_76  False  False  False  False  False  False  False  False
True
77  feature_77  False  False  False  False  False  False  False  False
True
78  feature_78  False  False  False  False  False  False  False  False
True
```

```
    tag_9  tag_10  tag_11  tag_12  tag_13  tag_14  tag_15  tag_16
0   False   False   False   False   False    True   False    True
1   False   False   False   False    True    True   False    True
2   False   False   False    True   False   False   False    True
3   False   False   False   False    True   False   False    True
4   False   False   False    True    True   False   False    True
..    ...     ...     ...     ...     ...     ...     ...     ...
74  False   False   False   False   False    True   False   False
75  False   False   False    True   False   False   False   False
76  False   False   False    True   False   False   False   False
77  False   False   False   False    True   False   False   False
78  False   False   False   False    True   False   False   False

[79 rows x 18 columns]
```

```
plt.figure(figsize=(20, 10))
plt.imshow(features.iloc[:, 1:].T.values, cmap="gray")
plt.xlabel("feature_00  ~  feature_78")
plt.ylabel("tag_0  ~  tag_16")
plt.yticks(np.arange(17))
plt.xticks(np.arange(79))
plt.grid()
plt.show()
```




```
# corr between feature_XX and feature_YY
plt.figure(figsize=(10, 10))
sns.heatmap(features[[ f"tag_{no}" for no in
range(0,17,1) ] ].T.corr(), square=True, cmap="jet")
plt.show()
```




# Responders

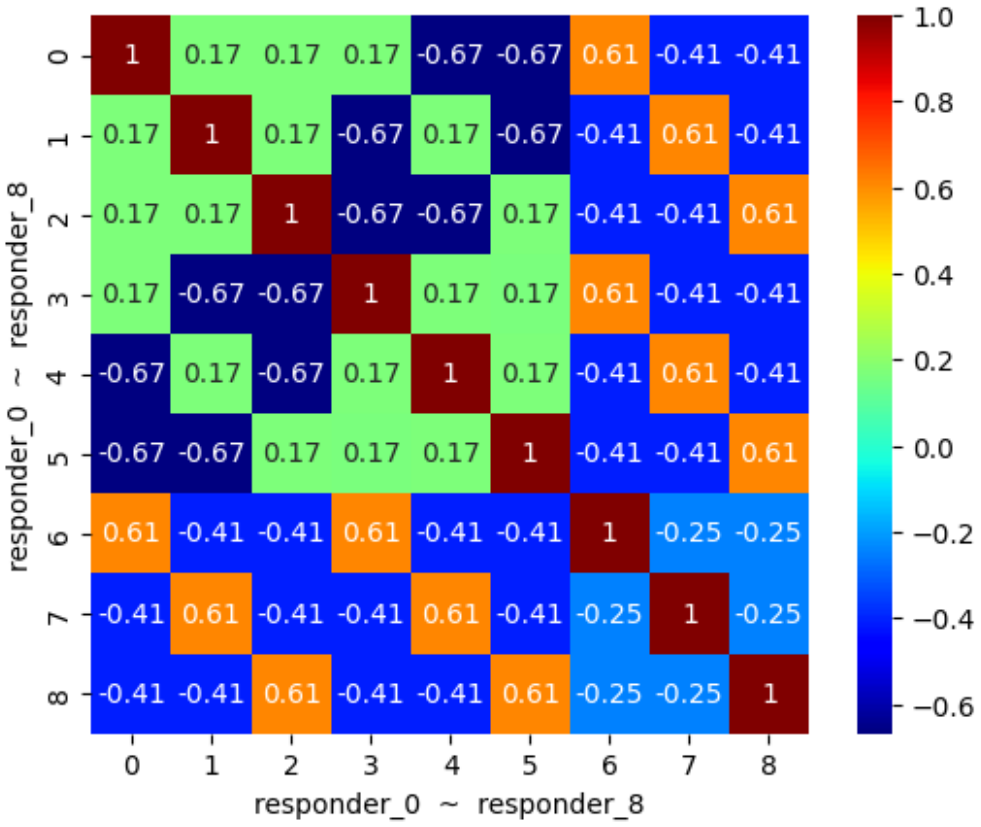
- responders.csv - metadata pertaining to the anonymized responders

```
responders = pd.read_csv(f"{ROOT_DIR}/responders.csv")
responders
```

| | responder | tag_0 | tag_1 | tag_2 | tag_3 | tag_4 |
|---|---|---|---|---|---|---|
| 0 | responder_0 | True | False | True | False | False |
| 1 | responder_1 | True | False | False | True | False |
| 2 | responder_2 | True | True | False | False | False |

```
3  responder_3  False  False   True  False   True
4  responder_4  False  False  False   True   True
5  responder_5  False   True  False  False   True
6  responder_6  False  False   True  False  False
7  responder_7  False  False  False   True  False
8  responder_8  False   True  False  False  False
```

```
# corr between responder_XX and responder_YY
sns.heatmap(responders[[ f"tag_{no}" for no in
range(0,5,1) ] ].T.corr(),  annot=True, square=True, cmap="jet")
plt.xlabel("responder_0  ~  responder_8")
plt.ylabel("responder_0  ~  responder_8")
plt.show()
```




# Sample submission

- **sample_submission.csv** - This file illustrates the format of the predictions your model should make.

```
sub = pd.read_csv(f"{ROOT_DIR}/sample_submission.csv")
print( f"sub.shape = {sub.shape}" )
sub
```

```
sub.shape = (39, 2)
```

| | row_id | responder_6 |
|---|---|---|
| 0 | 0 | 0.0 |
| 1 | 1 | 0.0 |
| 2 | 2 | 0.0 |
| 3 | 3 | 0.0 |
| 4 | 4 | 0.0 |
| 5 | 5 | 0.0 |
| 6 | 6 | 0.0 |
| 7 | 7 | 0.0 |
| 8 | 8 | 0.0 |
| 9 | 9 | 0.0 |
| 10 | 10 | 0.0 |
| 11 | 11 | 0.0 |
| 12 | 12 | 0.0 |
| 13 | 13 | 0.0 |
| 14 | 14 | 0.0 |
| 15 | 15 | 0.0 |
| 16 | 16 | 0.0 |
| 17 | 17 | 0.0 |
| 18 | 18 | 0.0 |
| 19 | 19 | 0.0 |
| 20 | 20 | 0.0 |
| 21 | 21 | 0.0 |
| 22 | 22 | 0.0 |
| 23 | 23 | 0.0 |
| 24 | 24 | 0.0 |
| 25 | 25 | 0.0 |
| 26 | 26 | 0.0 |
| 27 | 27 | 0.0 |
| 28 | 28 | 0.0 |
| 29 | 29 | 0.0 |
| 30 | 30 | 0.0 |
| 31 | 31 | 0.0 |
| 32 | 32 | 0.0 |
| 33 | 33 | 0.0 |
| 34 | 34 | 0.0 |
| 35 | 35 | 0.0 |
| 36 | 36 | 0.0 |
| 37 | 37 | 0.0 |
| 38 | 38 | 0.0 |

# Train.parquet

- **train.parquet** - The training set, contains historical data and returns. For convenience, the training set has been partitioned into ten parts.

- `date_id` and `time_id` - Integer values that are ordinally sorted, providing a chronological structure to the data, although the actual time intervals between `time_id` values may vary.
- `symbol_id` - Identifies a unique financial instrument.
- `weight` - The weighting used for calculating the scoring function.
- `feature_{00...78}` - Anonymized market data.
- `responder_{0...8}` - Anonymized responders clipped between -5 and 5. The responder_6 field is what you are trying to predict.

Each row in the `{train/test}.parquet` dataset corresponds to a unique combination of a symbol (identified by `symbol_id`) and a timestamp (represented by `date_id` and `time_id`). You will be provided with multiple responders, with `responder_6` being the only responder used for scoring. The date_id column is an integer which represents the day of the event, while time_id represents a time ordering. It's important to note that the real time differences between each time_id are not guaranteed to be consistent.

The `symbol_id` column contains encrypted identifiers. Each `symbol_id` is not guaranteed to appear in all `time_id` and `date_id` combinations. Additionally, new `symbol_id` values may appear in future test sets.

```
!tree /kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/
```

```
/kaggle/input/jane-street-real-time-market-data-forecasting/
train.parquet/
|-- partition_id=0
|   `-- part-0.parquet
|-- partition_id=1
|   `-- part-0.parquet
|-- partition_id=2
|   `-- part-0.parquet
|-- partition_id=3
|   `-- part-0.parquet
|-- partition_id=4
|   `-- part-0.parquet
|-- partition_id=5
|   `-- part-0.parquet
|-- partition_id=6
|   `-- part-0.parquet
|-- partition_id=7
|   `-- part-0.parquet
|-- partition_id=8
|   `-- part-0.parquet
`-- partition_id=9
    `-- part-0.parquet

10 directories, 10 files
```

```
train = (
    pl.read_parquet(f"{ROOT_DIR}/train.parquet/partition_id=0/part-
0.parquet")
)
train.shape
```

```
(1944210, 92)
```

```
train.head()
```

```
shape: (5, 92)
┌─────────┬─────────┬───────────┬──────────┬───┬─────────────┬─────────
──┬─────────────┬─────────────┐
│ date_id ┆ time_id ┆ symbol_id ┆ weight   ┆ … ┆ responder_ ┆
responder_ ┆ responder_ ┆ responder_ │
│ ---     ┆ ---     ┆ ---       ┆ ---      ┆   ┆ 5           ┆ 6
┆ 7           ┆ 8           │
│ i16     ┆ i16     ┆ i8        ┆ f32      ┆   ┆ ---         ┆ ---
┆ ---         ┆ ---         │
│         ┆         ┆           ┆          ┆   ┆ f32         ┆ f32
┆ f32         ┆ f32         │
╞═════════╪═════════╪═══════════╪══════════╪═══╪═════════════╪═══════
══╪═════════════╪═════════════╡
│ 0       ┆ 0       ┆ 1         ┆ 3.889038 ┆ … ┆ 1.218368    ┆ 0.775981
┆ 0.346999    ┆ 0.095504    │
│ 0       ┆ 0       ┆ 7         ┆ 1.370613 ┆ … ┆ 5.0         ┆ 0.703665
┆ 0.216683    ┆ 0.778639    │
│ 0       ┆ 0       ┆ 9         ┆ 2.285698 ┆ … ┆ 0.099793    ┆ 2.109352
┆ 0.670881    ┆ 0.772828    │
│ 0       ┆ 0       ┆ 10        ┆ 0.690606 ┆ … ┆ 1.225376    ┆ 1.114137
┆ 0.775199    ┆ -1.379516   │
│ 0       ┆ 0       ┆ 14        ┆ 0.44057  ┆ … ┆ -5.0        ┆ -3.57282
┆ -1.089123   ┆ -5.0        │
└─────────┴─────────┴───────────┴──────────┴───┴─────────────┴─────────
──┴─────────────┴─────────────┘
```

```
print(str(train.columns))
```

```
['date_id', 'time_id', 'symbol_id', 'weight', 'feature_00',
'feature_01', 'feature_02', 'feature_03', 'feature_04', 'feature_05',
'feature_06', 'feature_07', 'feature_08', 'feature_09', 'feature_10',
'feature_11', 'feature_12', 'feature_13', 'feature_14', 'feature_15',
'feature_16', 'feature_17', 'feature_18', 'feature_19', 'feature_20',
'feature_21', 'feature_22', 'feature_23', 'feature_24', 'feature_25',
'feature_26', 'feature_27', 'feature_28', 'feature_29', 'feature_30',
'feature_31', 'feature_32', 'feature_33', 'feature_34', 'feature_35',
'feature_36', 'feature_37', 'feature_38', 'feature_39', 'feature_40',
'feature_41', 'feature_42', 'feature_43', 'feature_44', 'feature_45',
'feature_46', 'feature_47', 'feature_48', 'feature_49', 'feature_50',
'feature_51', 'feature_52', 'feature_53', 'feature_54', 'feature_55',
```

```
'feature_56', 'feature_57', 'feature_58', 'feature_59', 'feature_60',
'feature_61', 'feature_62', 'feature_63', 'feature_64', 'feature_65',
'feature_66', 'feature_67', 'feature_68', 'feature_69', 'feature_70',
'feature_71', 'feature_72', 'feature_73', 'feature_74', 'feature_75',
'feature_76', 'feature_77', 'feature_78', 'responder_0',
'responder_1', 'responder_2', 'responder_3', 'responder_4',
'responder_5', 'responder_6', 'responder_7', 'responder_8']
```

```
train.schema
```

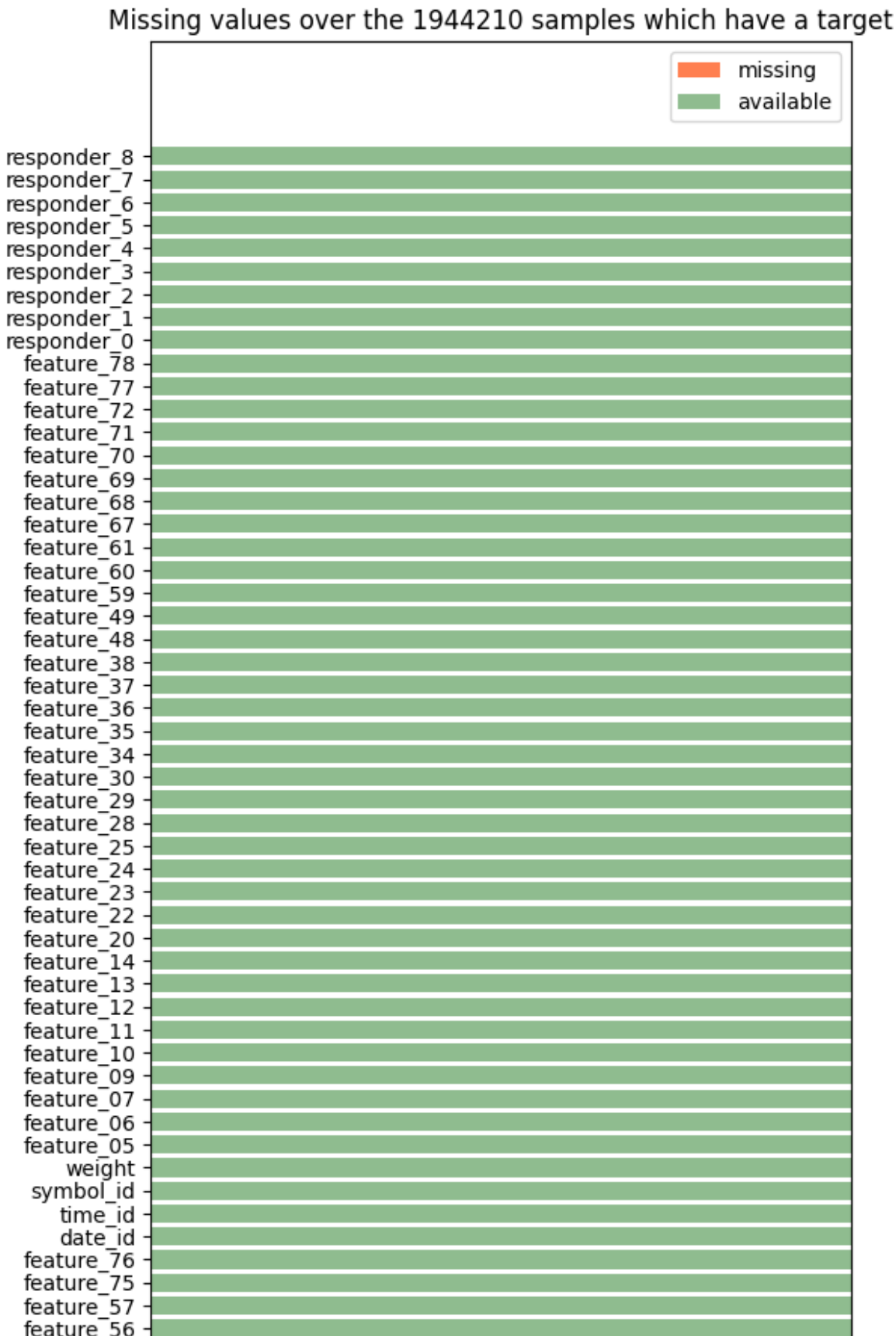
```
Schema([('date_id', Int16),
        ('time_id', Int16),
        ('symbol_id', Int8),
        ('weight', Float32),
        ('feature_00', Float32),
        ('feature_01', Float32),
        ('feature_02', Float32),
        ('feature_03', Float32),
        ('feature_04', Float32),
        ('feature_05', Float32),
        ('feature_06', Float32),
        ('feature_07', Float32),
        ('feature_08', Float32),
        ('feature_09', Int8),
        ('feature_10', Int8),
        ('feature_11', Int16),
        ('feature_12', Float32),
        ('feature_13', Float32),
        ('feature_14', Float32),
        ('feature_15', Float32),
        ('feature_16', Float32),
        ('feature_17', Float32),
        ('feature_18', Float32),
        ('feature_19', Float32),
        ('feature_20', Float32),
        ('feature_21', Float32),
        ('feature_22', Float32),
        ('feature_23', Float32),
        ('feature_24', Float32),
        ('feature_25', Float32),
        ('feature_26', Float32),
        ('feature_27', Float32),
        ('feature_28', Float32),
        ('feature_29', Float32),
        ('feature_30', Float32),
        ('feature_31', Float32),
        ('feature_32', Float32),
        ('feature_33', Float32),
        ('feature_34', Float32),
        ('feature_35', Float32),
```

```
        ('feature_36', Float32),
        ('feature_37', Float32),
        ('feature_38', Float32),
        ('feature_39', Float32),
        ('feature_40', Float32),
        ('feature_41', Float32),
        ('feature_42', Float32),
        ('feature_43', Float32),
        ('feature_44', Float32),
        ('feature_45', Float32),
        ('feature_46', Float32),
        ('feature_47', Float32),
        ('feature_48', Float32),
        ('feature_49', Float32),
        ('feature_50', Float32),
        ('feature_51', Float32),
        ('feature_52', Float32),
        ('feature_53', Float32),
        ('feature_54', Float32),
        ('feature_55', Float32),
        ('feature_56', Float32),
        ('feature_57', Float32),
        ('feature_58', Float32),
        ('feature_59', Float32),
        ('feature_60', Float32),
        ('feature_61', Float32),
        ('feature_62', Float32),
        ('feature_63', Float32),
        ('feature_64', Float32),
        ('feature_65', Float32),
        ('feature_66', Float32),
        ('feature_67', Float32),
        ('feature_68', Float32),
        ('feature_69', Float32),
        ('feature_70', Float32),
        ('feature_71', Float32),
        ('feature_72', Float32),
        ('feature_73', Float32),
        ('feature_74', Float32),
        ('feature_75', Float32),
        ('feature_76', Float32),
        ('feature_77', Float32),
        ('feature_78', Float32),
        ('responder_0', Float32),
        ('responder_1', Float32),
        ('responder_2', Float32),
        ('responder_3', Float32),
        ('responder_4', Float32),
        ('responder_5', Float32),
```

```
        ('responder_6', Float32),
        ('responder_7', Float32),
        ('responder_8', Float32)])
```

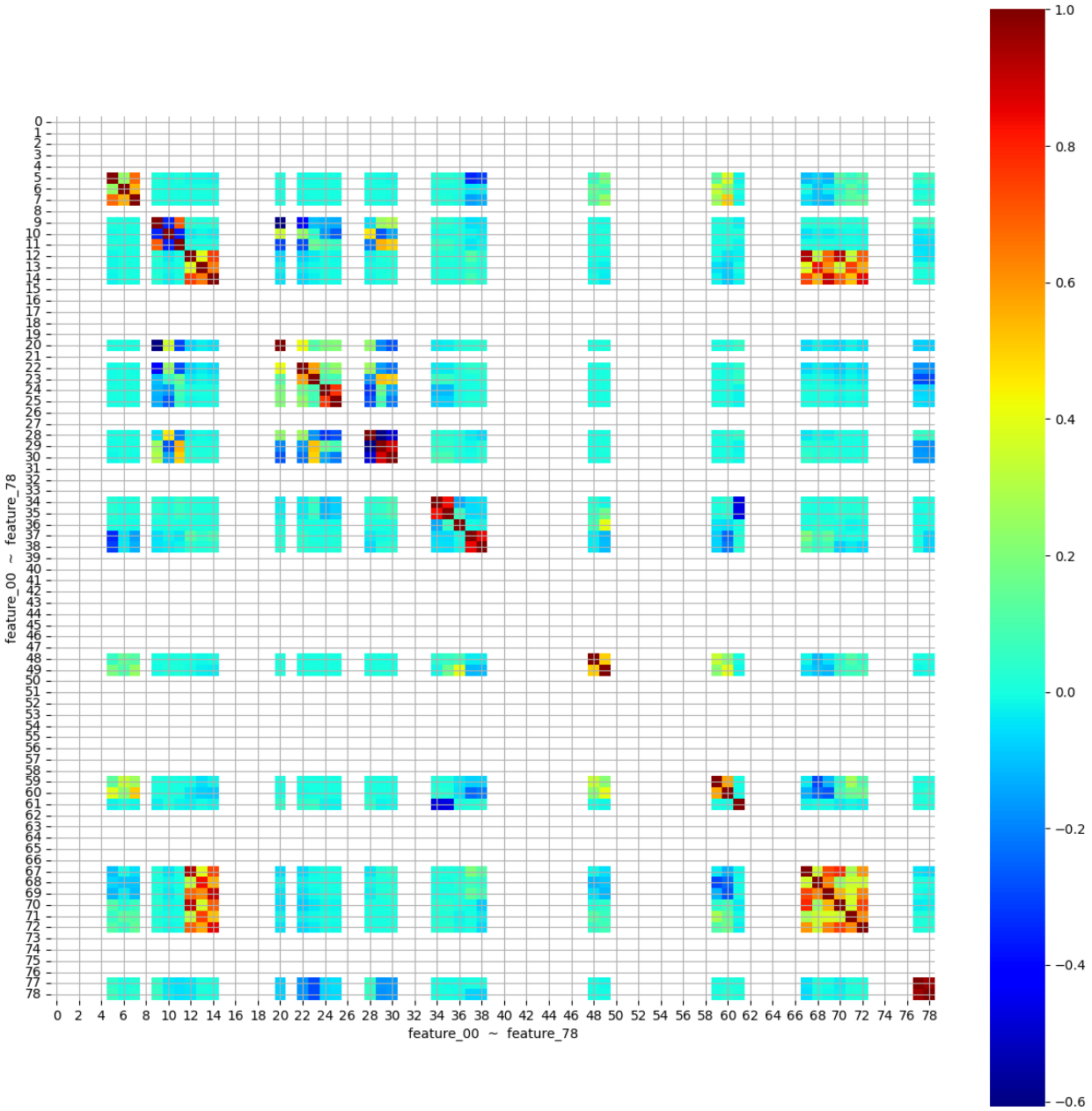
## Missing values

```
# only look at the data where responder_6 is not null
supervised_usable = (
    train
    .filter(pl.col('responder_6').is_not_null())
)

missing_count = (
    supervised_usable
    .null_count()  # Counts null values in each column
    .transpose(
        include_header=True,
        header_name='feature',
        column_names=['null_count']
    )  # Transposes the DataFrame, making columns into rows
    .sort('null_count', descending=True)  # Sorts by number of nulls
(highest to lowest)
    .with_columns(
        (pl.col('null_count') /
len(supervised_usable)).alias('null_ratio')
    )  # Adds a new column showing the ratio of nulls
)

plt.figure(figsize=(6, 20))
plt.title(f'Missing values over the {len(supervised_usable)} samples
which have a target')
plt.barh(np.arange(len(missing_count)),
missing_count.get_column('null_ratio'), color='coral',
label='missing')
plt.barh(np.arange(len(missing_count)),
         1 - missing_count.get_column('null_ratio'),
         left=missing_count.get_column('null_ratio'),
         color='darkseagreen', label='available')
plt.yticks(np.arange(len(missing_count)),
missing_count.get_column('feature'))
plt.gca().xaxis.set_major_formatter(PercentFormatter(xmax=1,
decimals=0))
plt.xlim(0, 1)
plt.legend()
plt.show()
```


Missing values over the 1944210 samples which have a target
missing
available
responder_8
responder_7
responder_6
responder_5
responder_4
responder_3
responder_2
responder_1
responder_0
feature_78
feature_77
feature_72
feature_71
feature_70
feature_69
feature_68
feature_67
feature_61
feature_60
feature_59
feature_49
feature_48
feature_38
feature_37
feature_36
feature_35
feature_34
feature_30
feature_29
feature_28
feature_25
feature_24
feature_23
feature_22
feature_20
feature_14
feature_13
feature_12
feature_11
feature_10
feature_09
feature_07
feature_06
feature_05
weight
symbol_id
time_id
date_id
feature_76
feature_75
feature_57
feature_56

# feature_00-78

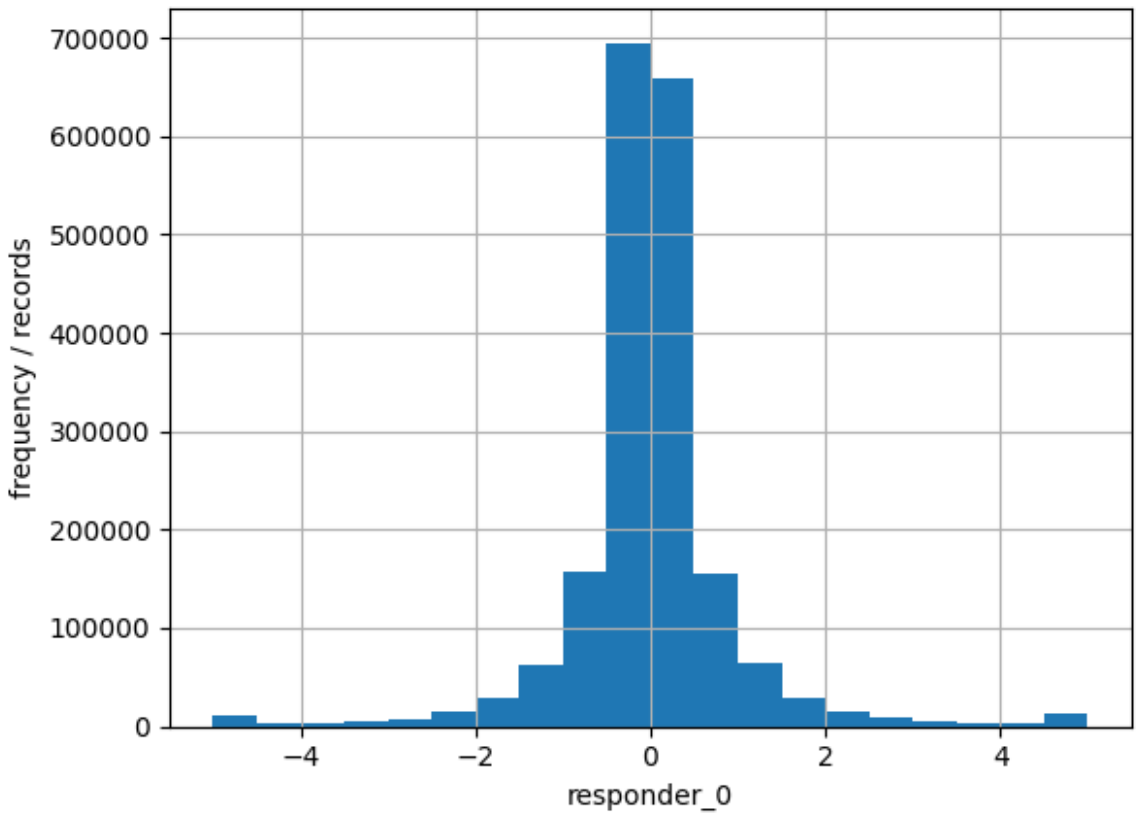
```
plt.figure(figsize=(15, 15))
sns.heatmap(train[[ f"feature_{target:02d}" for target in
range(79)]].corr(), square=True, cmap="jet")
plt.xlabel("feature_00  ~  feature_78")
plt.ylabel("feature_00  ~  feature_78")
plt.grid()
plt.show()
```



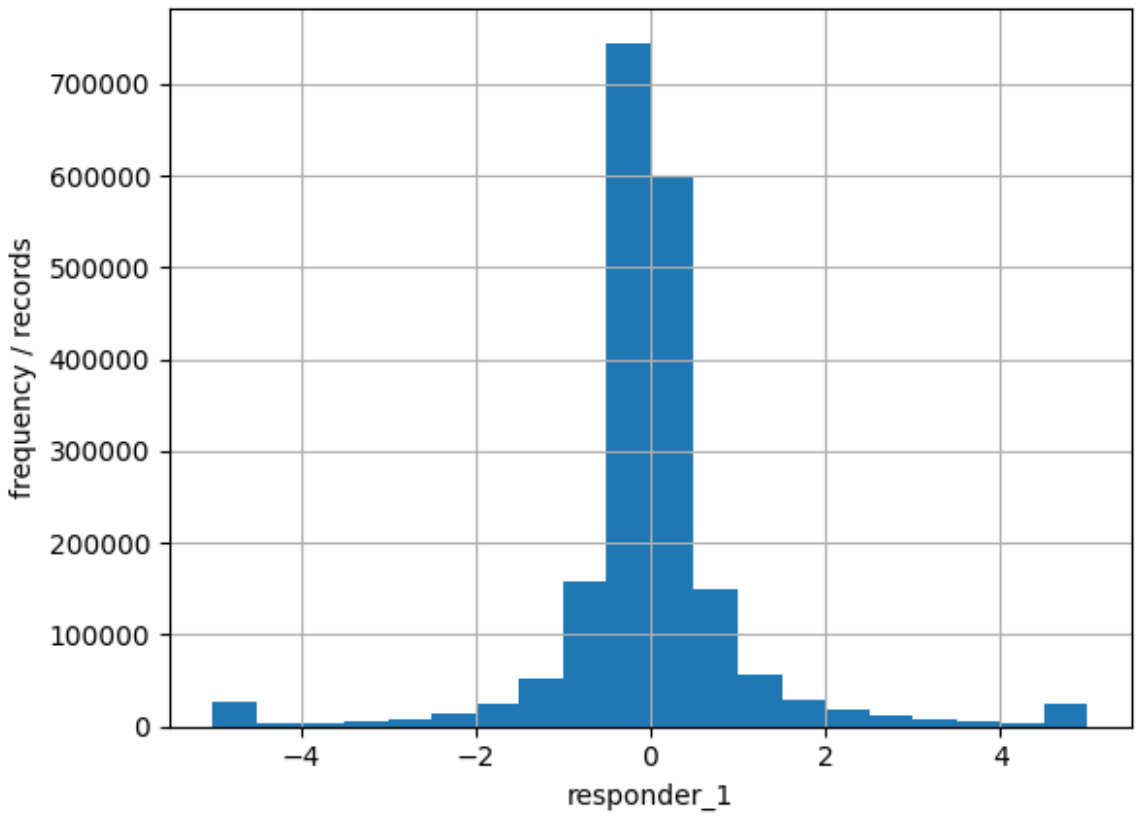
## responder_0 - 8

```
for target in range(9):
    col = f"responder_{target}"
    mean_, sgm_ = train[col].mean(), np.sqrt(train[col].var())
    min_, max_ = train[col].min(), train[col].max()
    print("- " * 30)
    print( f"column = {col}" )
    print( f" - mean  : {mean_:.4f}",  )
    print( f" - sigma : {sgm_:.4f}",  )
    print( f" - min   : {min_:.4f}",  )
    print( f" - max   : {max_:.4f}",  )

    plt.hist(train[col], bins=20)
    plt.xlabel(col)
    plt.ylabel("frequency / records")
    #plt.yscale("log")
    plt.grid()
    plt.show()
```

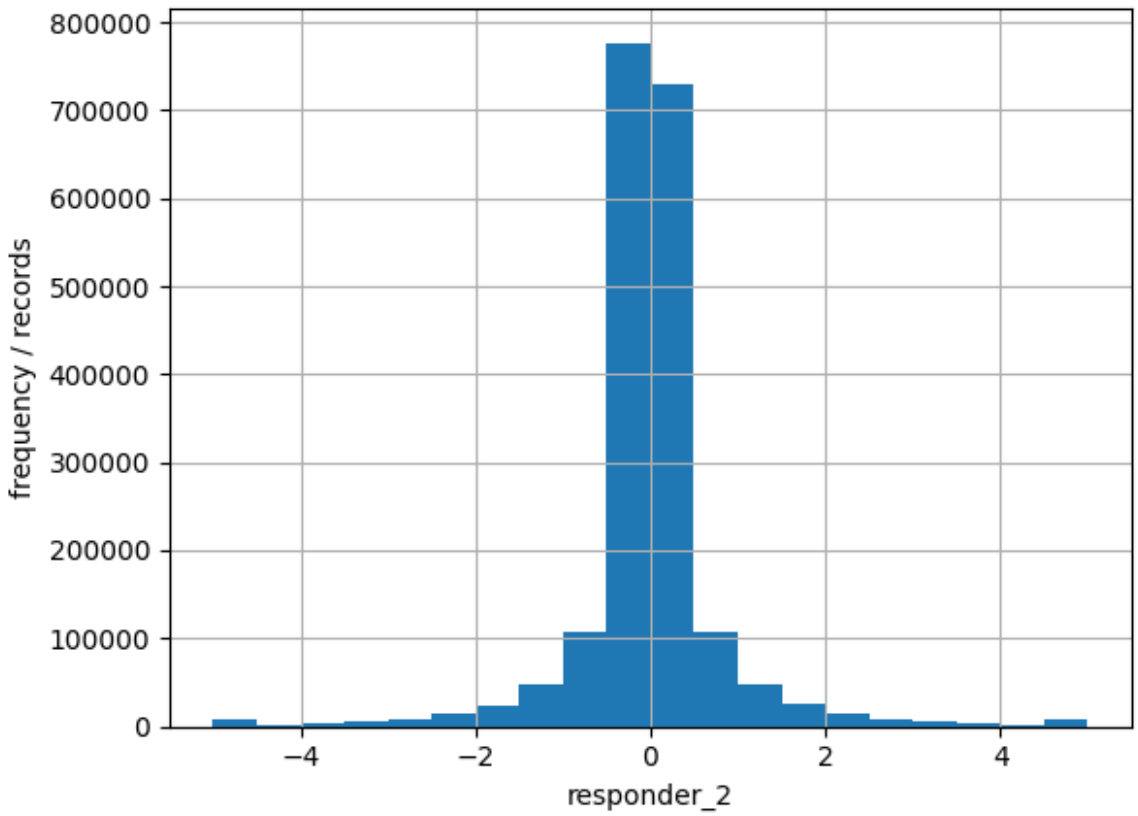
```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_0
 - mean  : 0.0084
 - sigma : 0.9559
 - min   : -5.0000
 - max   : 5.0000
```



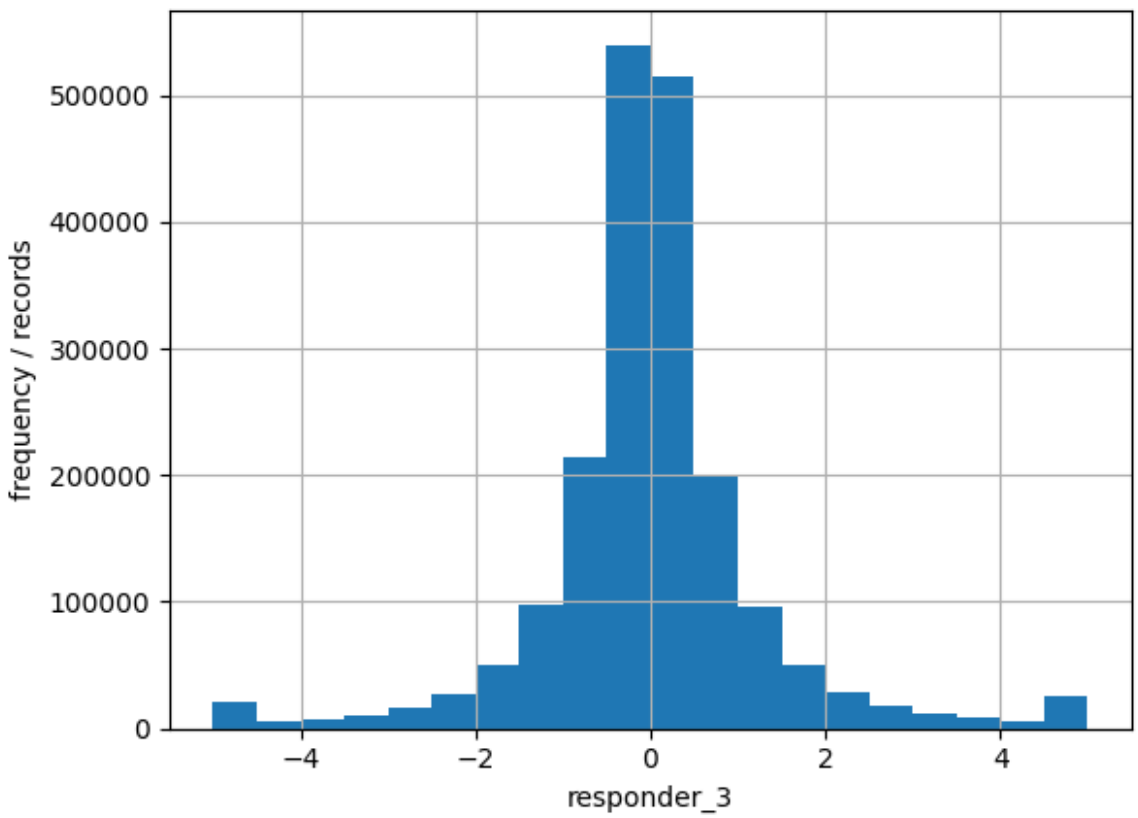

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_1
 - mean  : 0.0108
 - sigma : 1.1418
 - min   : -5.0000
 - max   : 5.0000
```



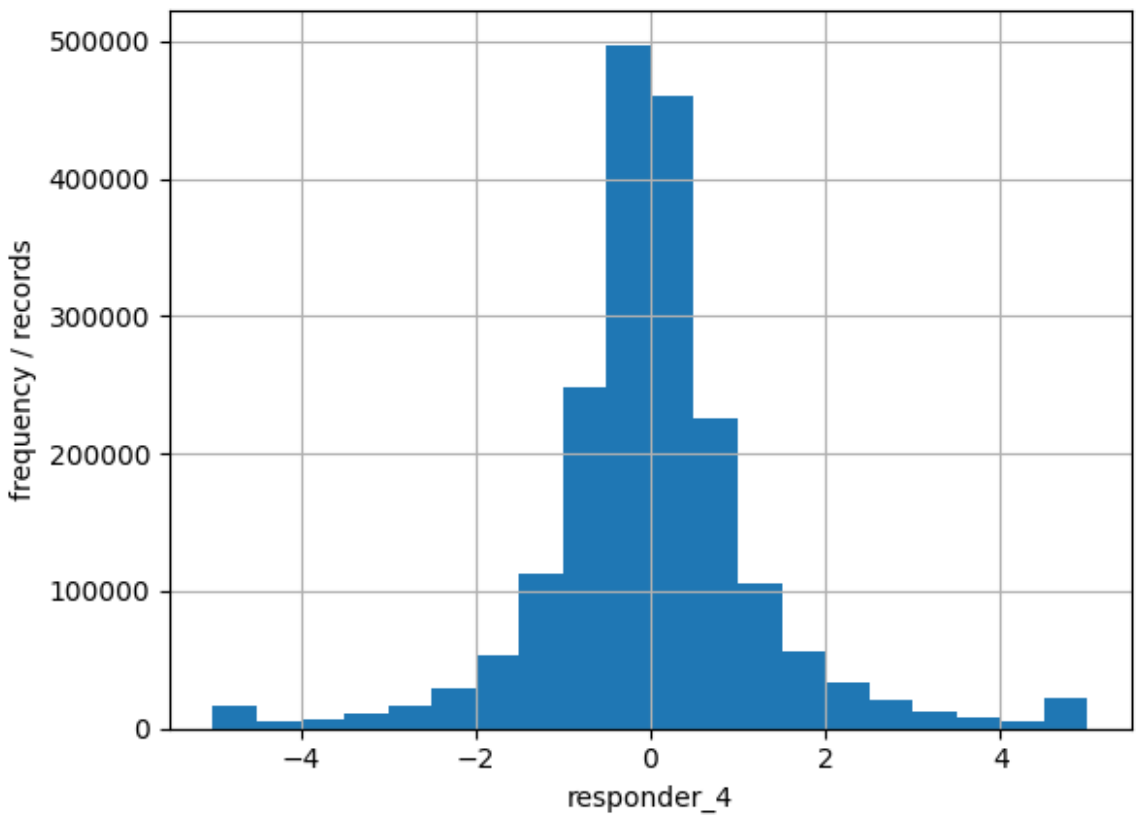

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_2
 - mean  : 0.0024
 - sigma : 0.8442
 - min   : -5.0000
 - max   : 5.0000
```



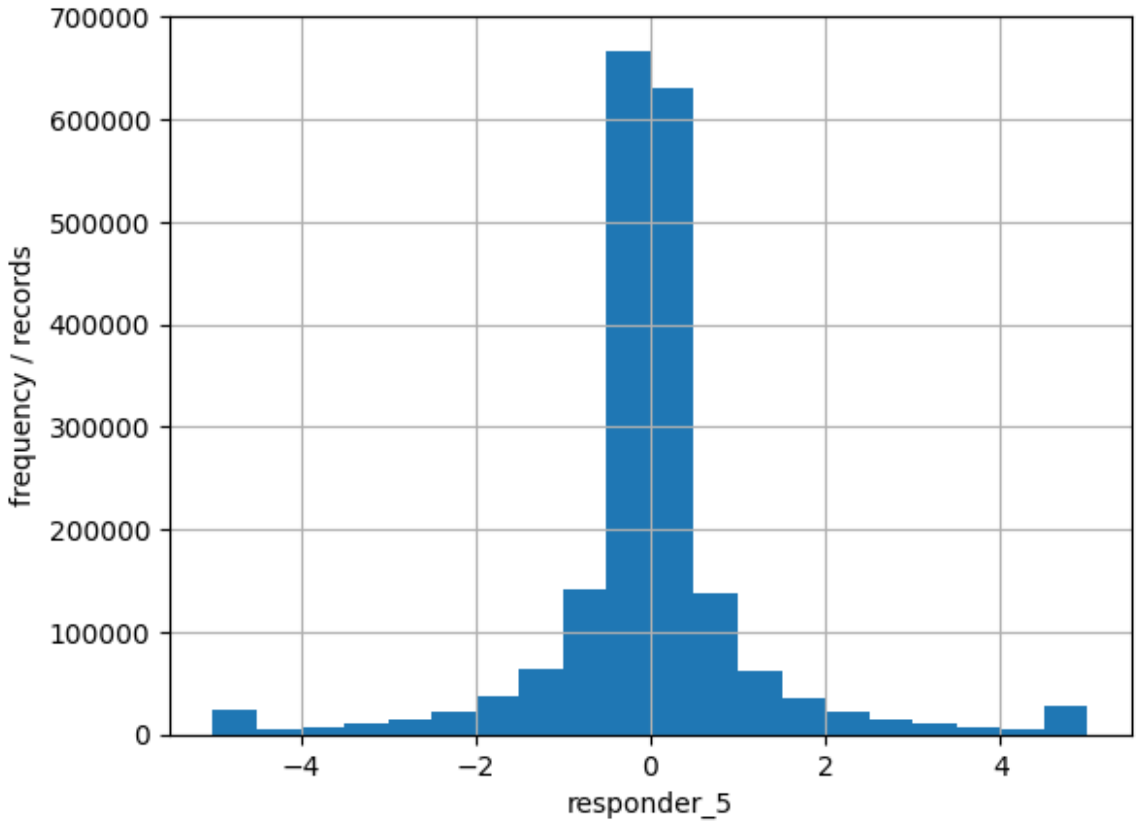

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_3
 - mean  : 0.0114
 - sigma : 1.2760
 - min   : -5.0000
 - max   : 5.0000
```



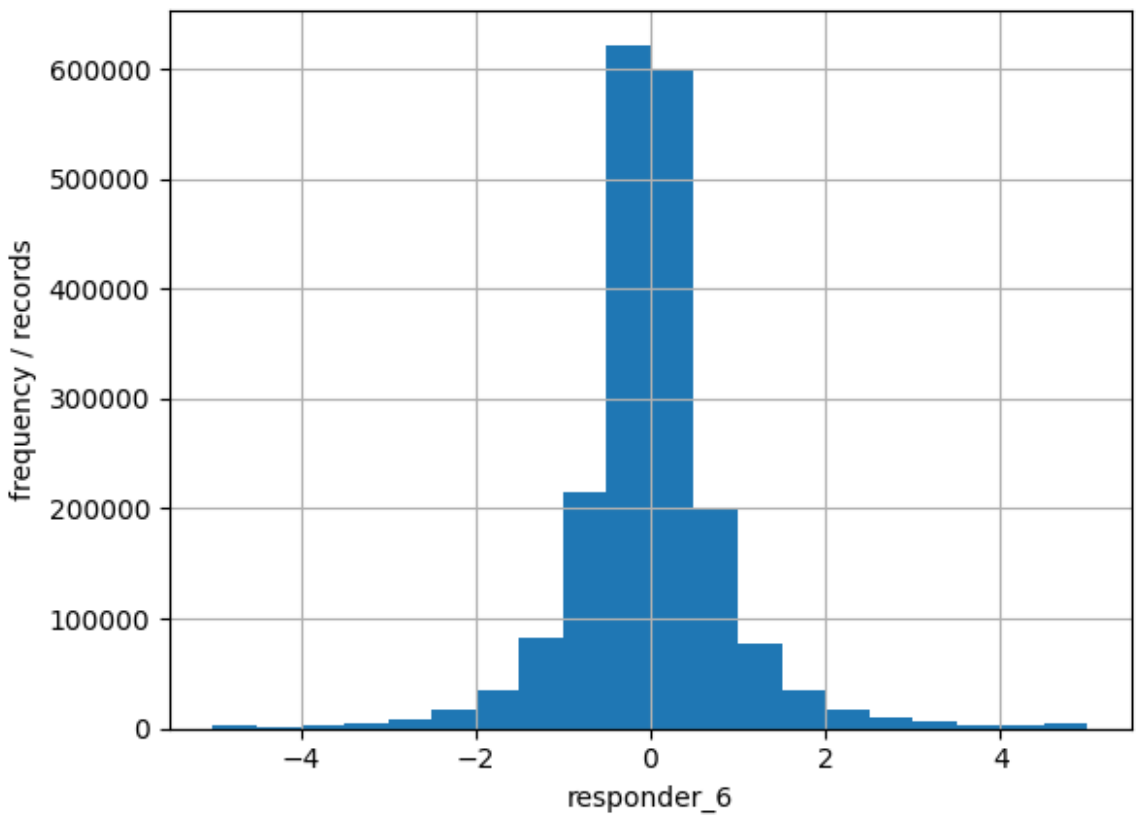

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_4
 - mean  : 0.0219
 - sigma : 1.2664
 - min   : -5.0000
 - max   : 5.0000
```



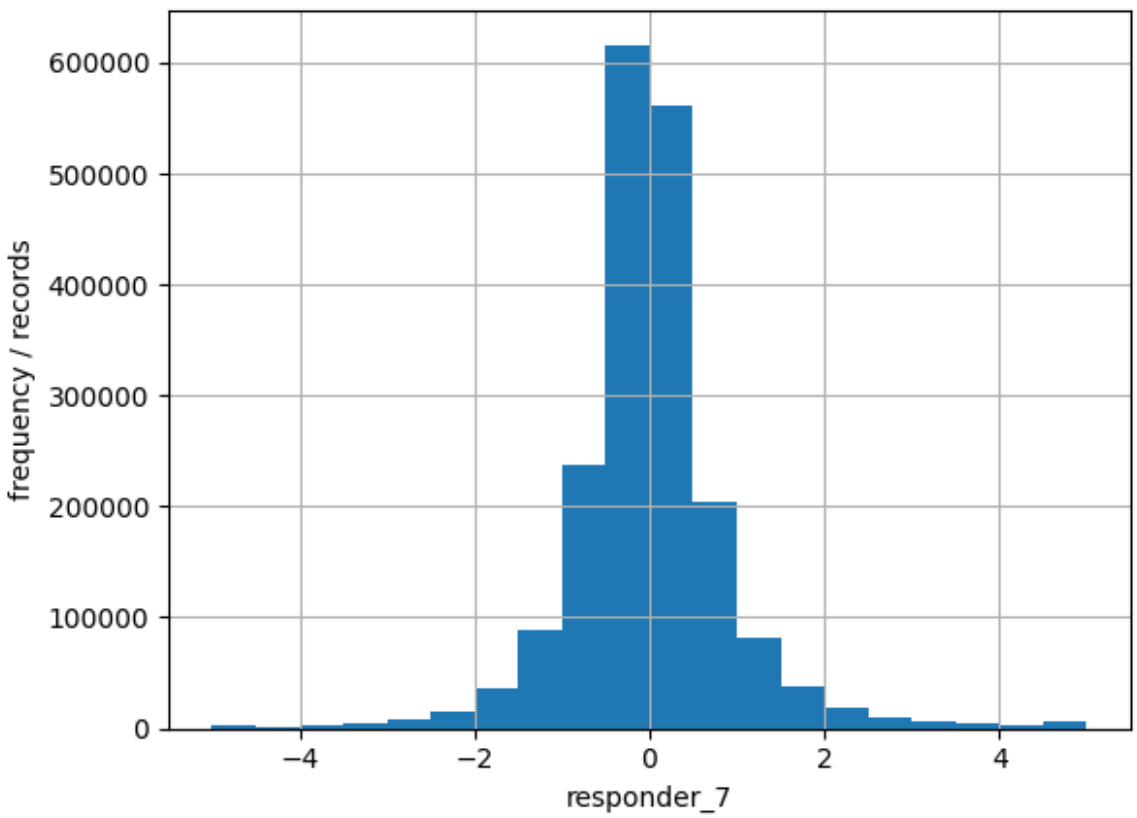

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_5
 - mean  : 0.0033
 - sigma : 1.2252
 - min   : -5.0000
 - max   : 5.0000
```



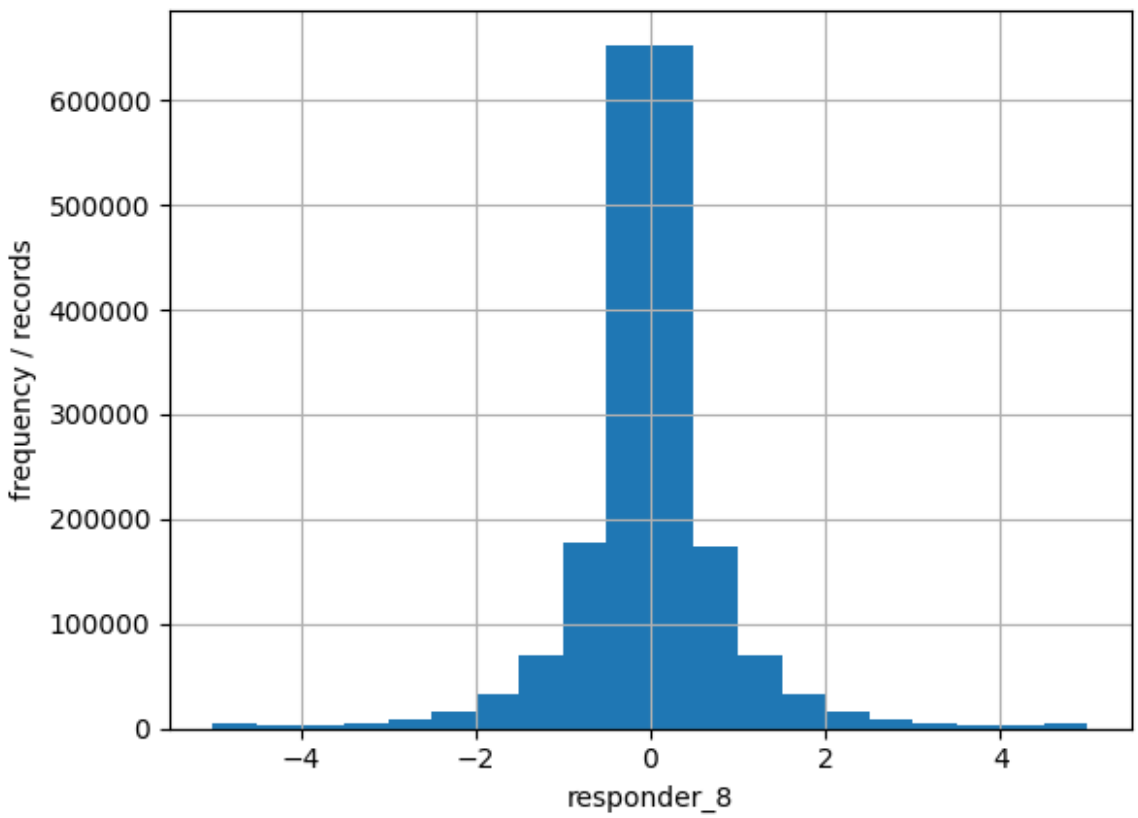

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_6
 - mean  : 0.0015
 - sigma : 0.8706
 - min   : -5.0000
 - max   : 5.0000
```



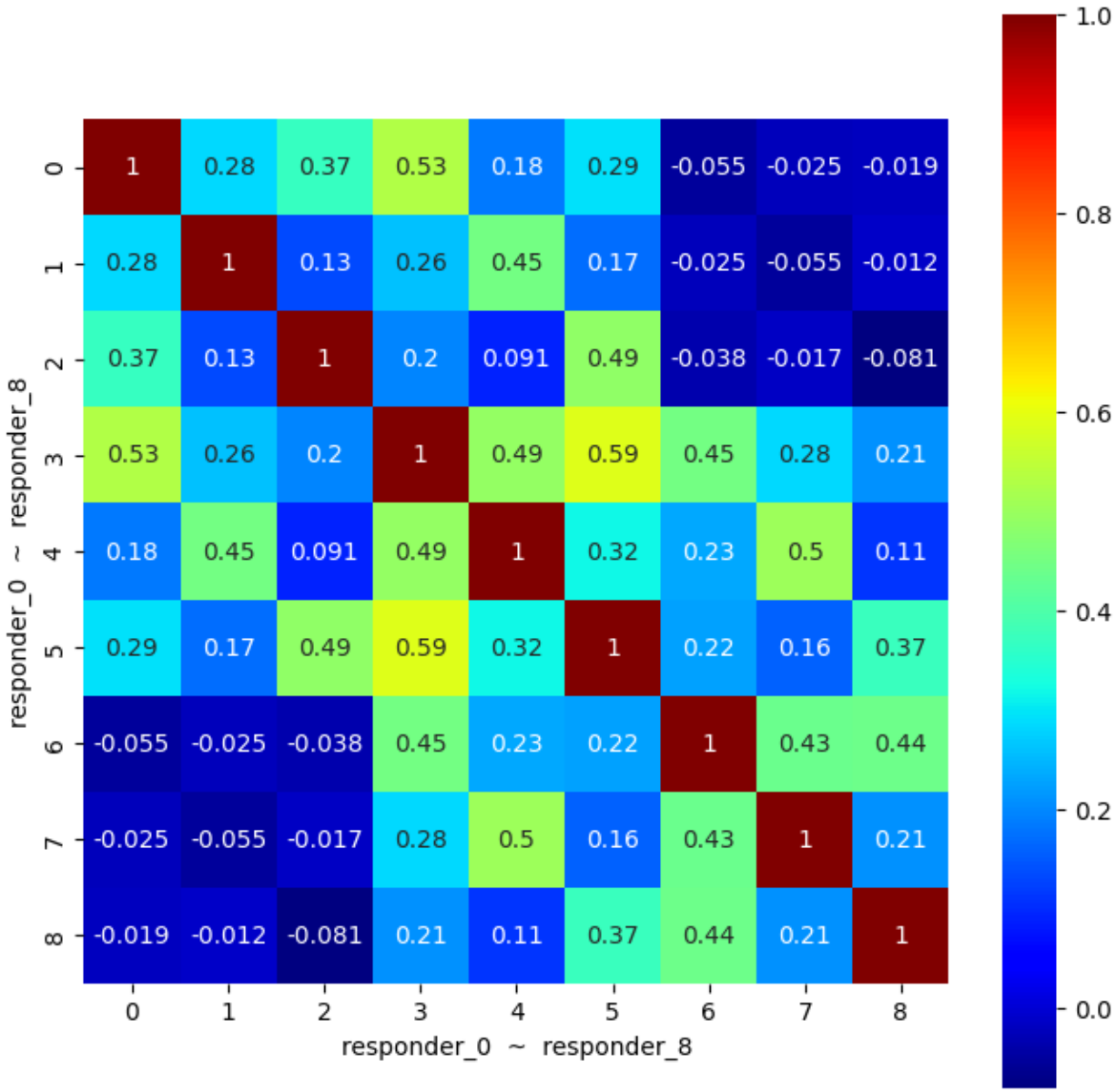

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_7
 - mean  : -0.0005
 - sigma : 0.8918
 - min   : -5.0000
 - max   : 5.0000
```




```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
column = responder_8
 - mean  : 0.0011
 - sigma : 0.8737
 - min   : -5.0000
 - max   : 5.0000
```



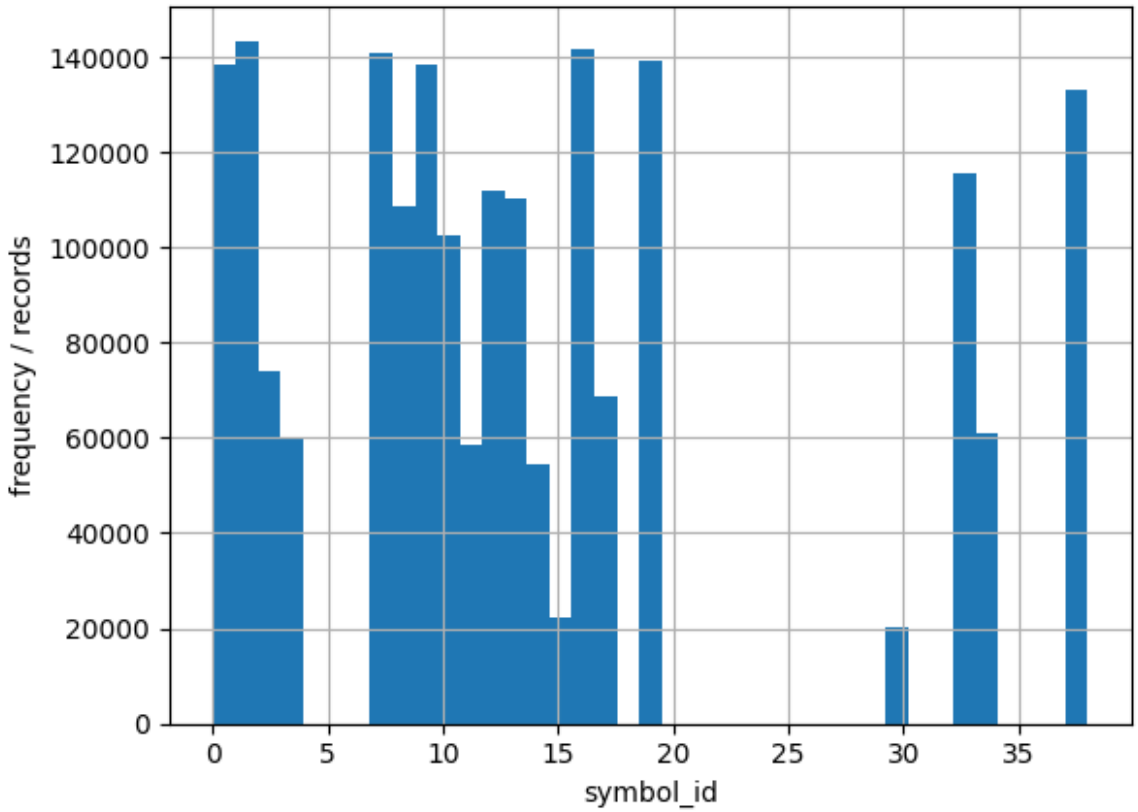

```
plt.figure(figsize=(8, 8))
sns.heatmap(train[[ f"responder_{target}" for target in
range(9)]].corr(),  annot=True, square=True, cmap="jet")
plt.xlabel("responder_0  ~  responder_8")
plt.ylabel("responder_0  ~  responder_8")
plt.show()
```



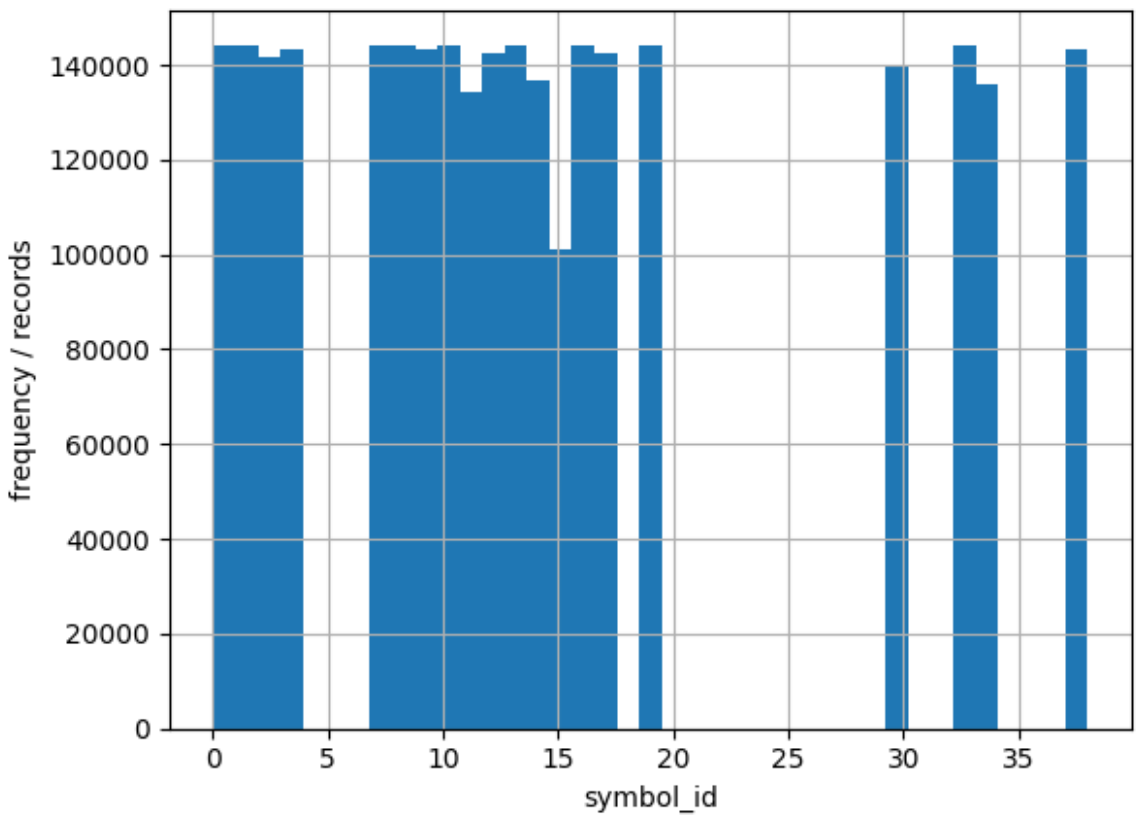

## symbol_id

```
for partition_id in range(10):
    print(f"> train.parquet/partition_id={partition_id}/part-
0.parquet")
    train_data =
pl.read_parquet(f"{ROOT_DIR}/train.parquet/partition_id={partition_id}
/part-0.parquet")

    print( f"symbol_id: ", train_data["symbol_id"].min(), "-",
train_data["symbol_id"].max())
    bins = train_data["symbol_id"].max() -
train_data["symbol_id"].min() + 1
```

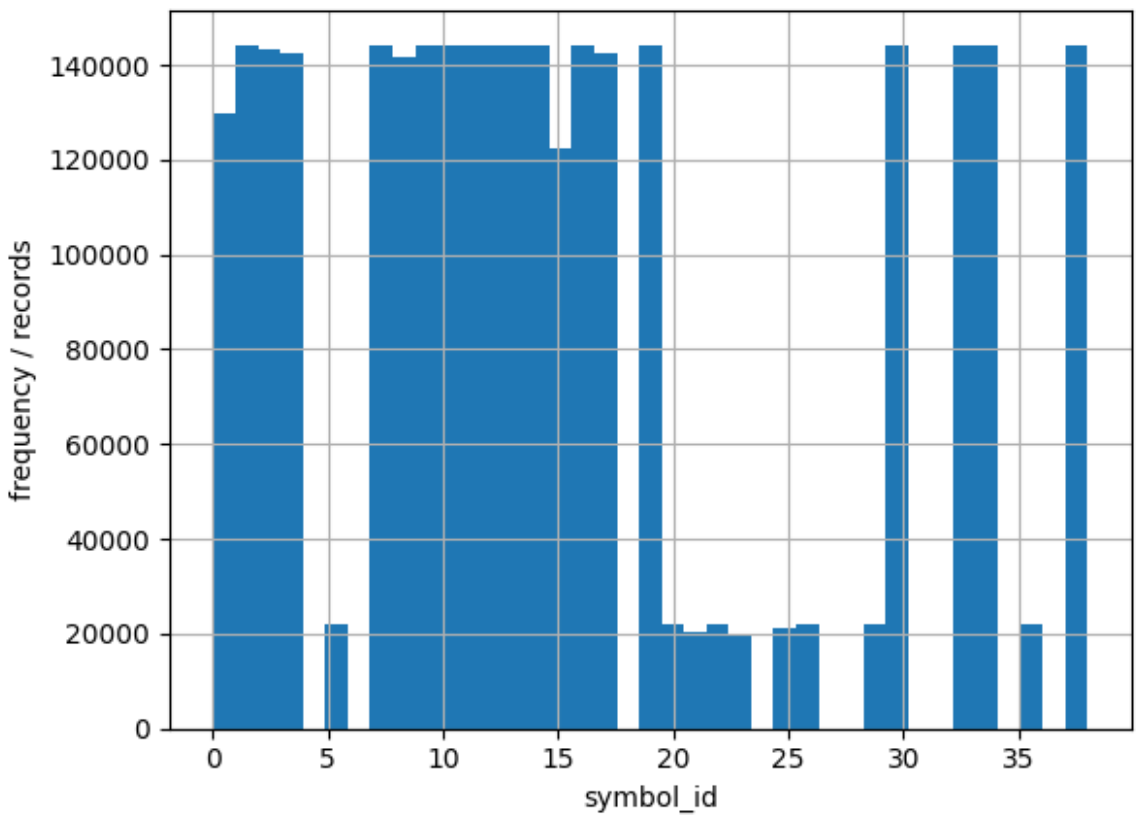
```
    plt.hist(train_data["symbol_id"], bins=bins)
    plt.xlabel("symbol_id")
    plt.ylabel("frequency / records")
    plt.grid()
    plt.show()

> train.parquet/partition_id=0/part-0.parquet
symbol_id:  0 - 38
```



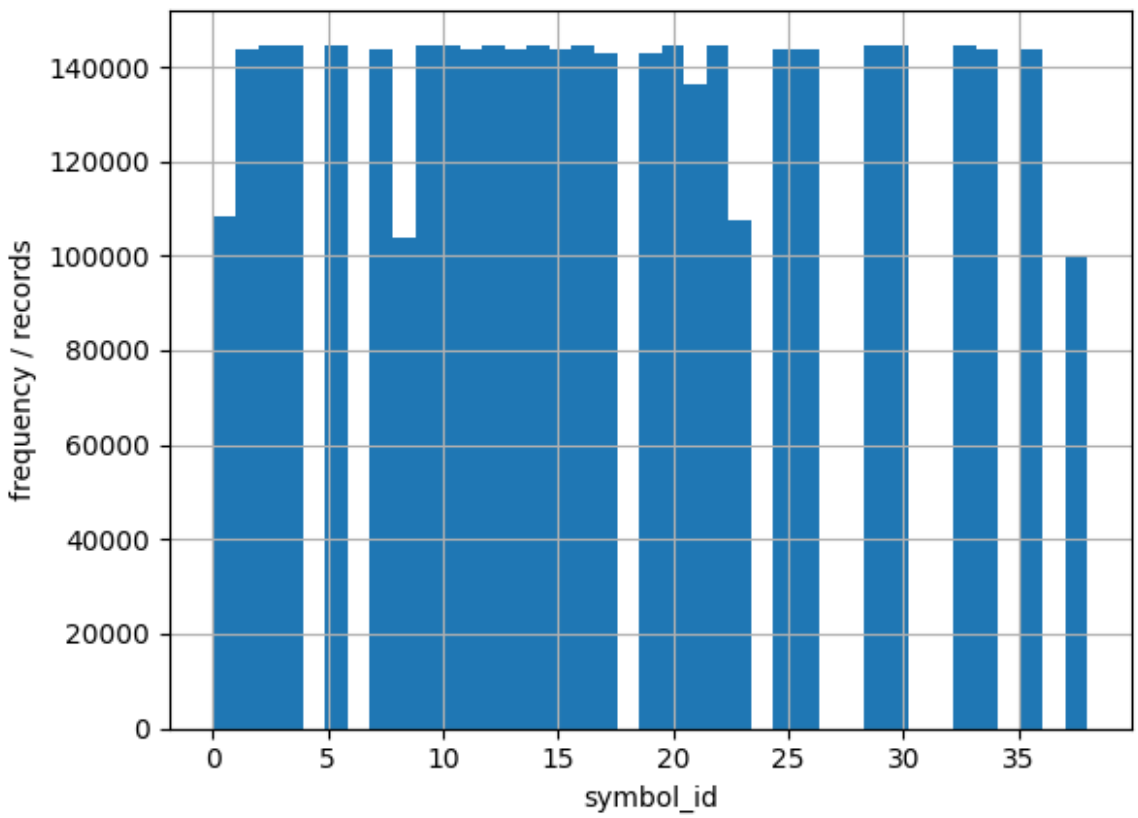

```
> train.parquet/partition_id=1/part-0.parquet
symbol_id:  0 - 38
```



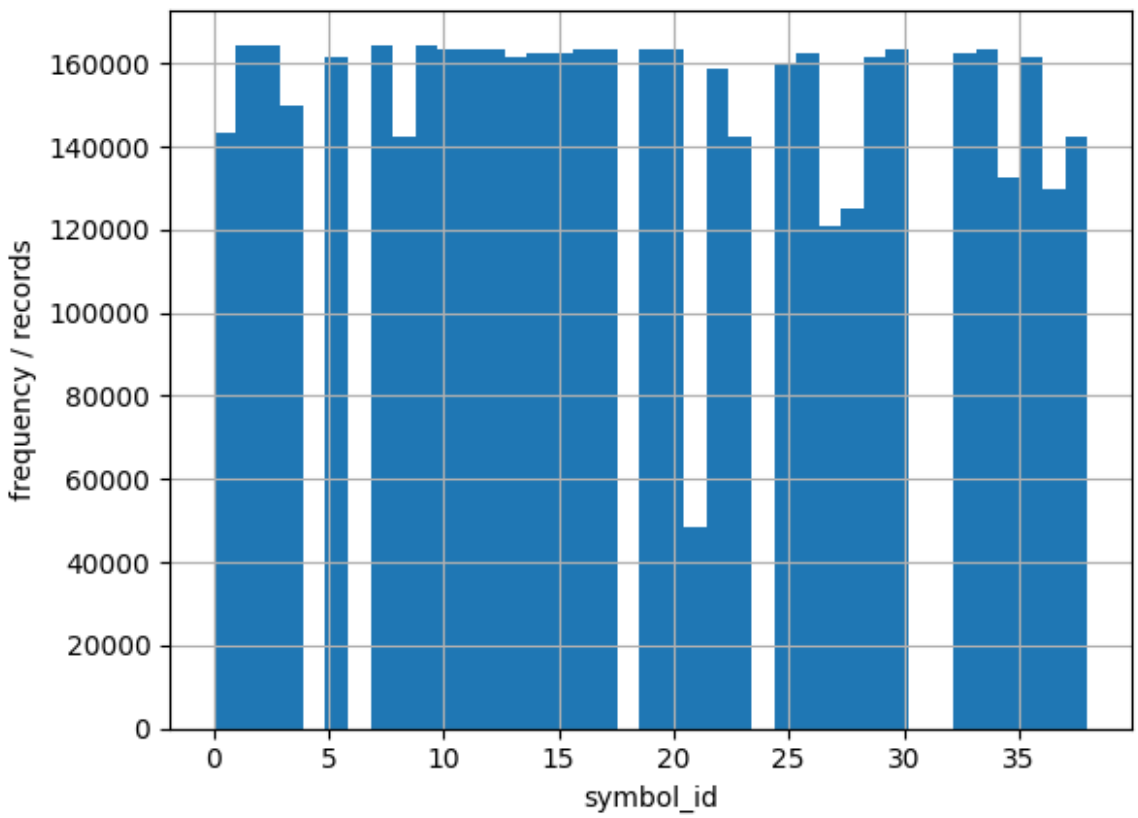

```
> train.parquet/partition_id=2/part-0.parquet
symbol_id:  0 - 38
```



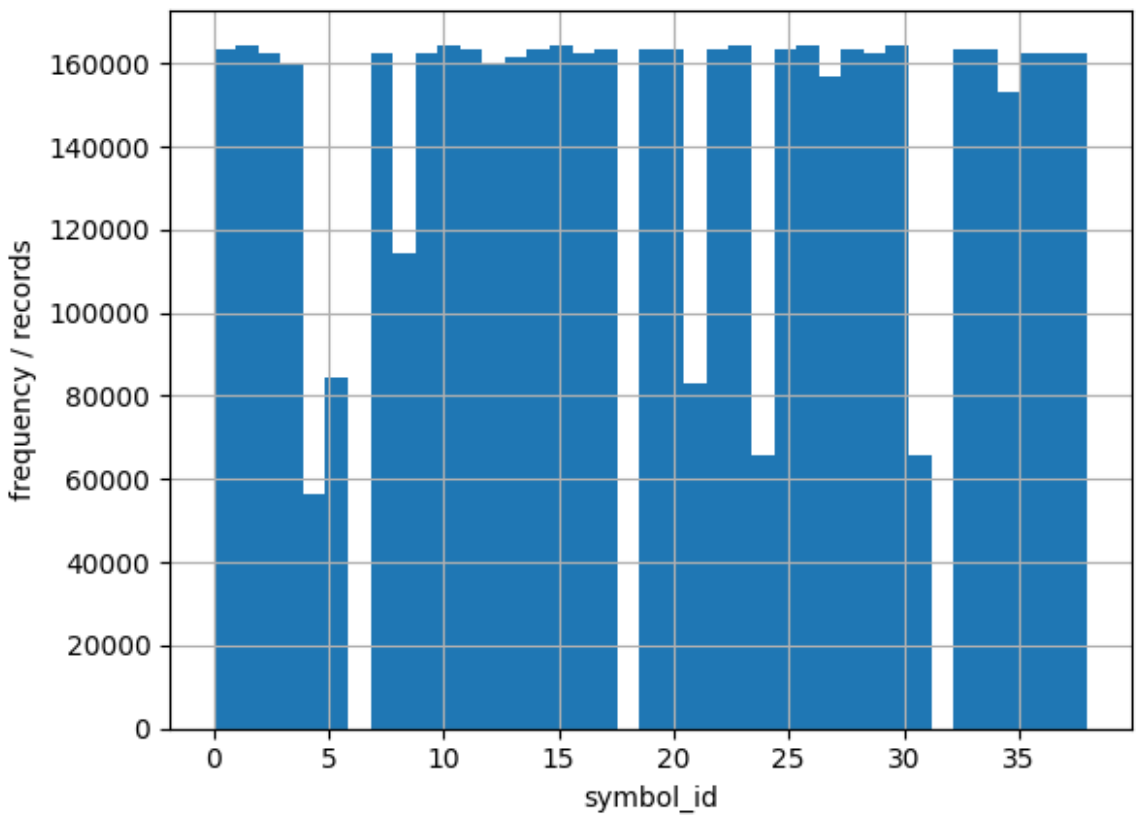

```
> train.parquet/partition_id=3/part-0.parquet
symbol_id:  0 - 38
```



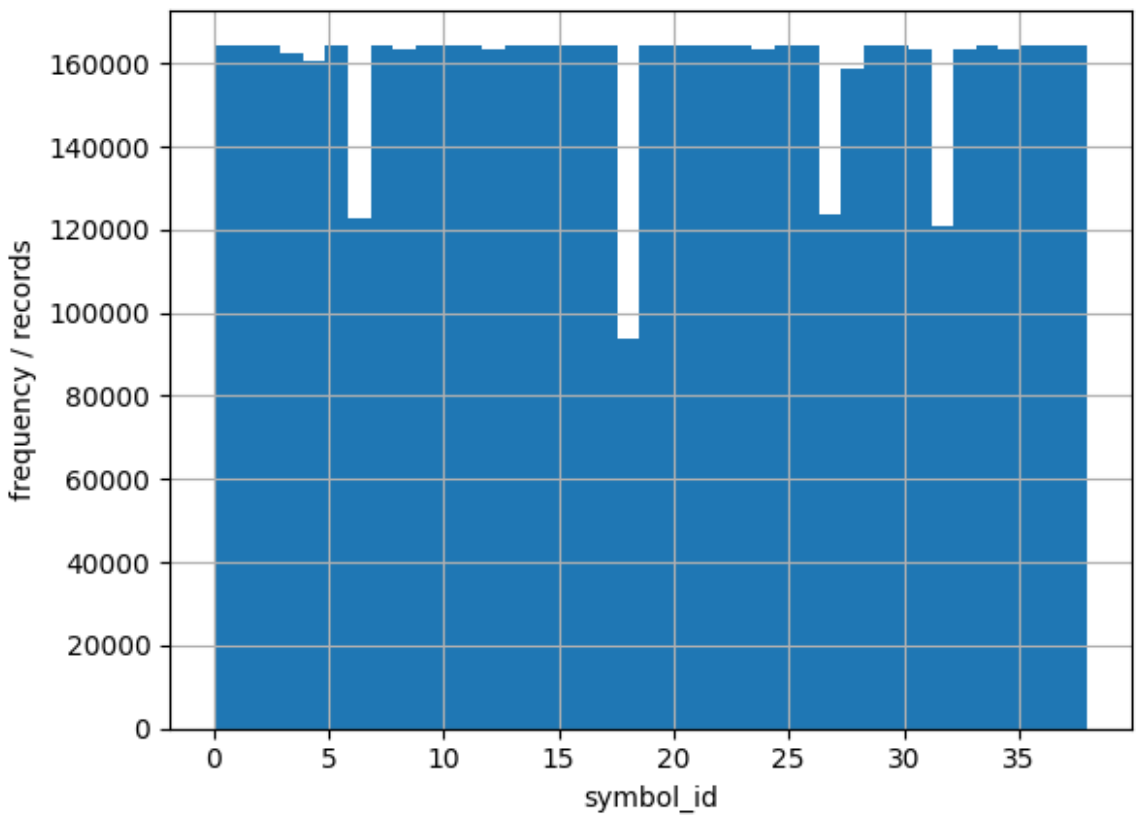

```
> train.parquet/partition_id=4/part-0.parquet
symbol_id:  0 - 38
```



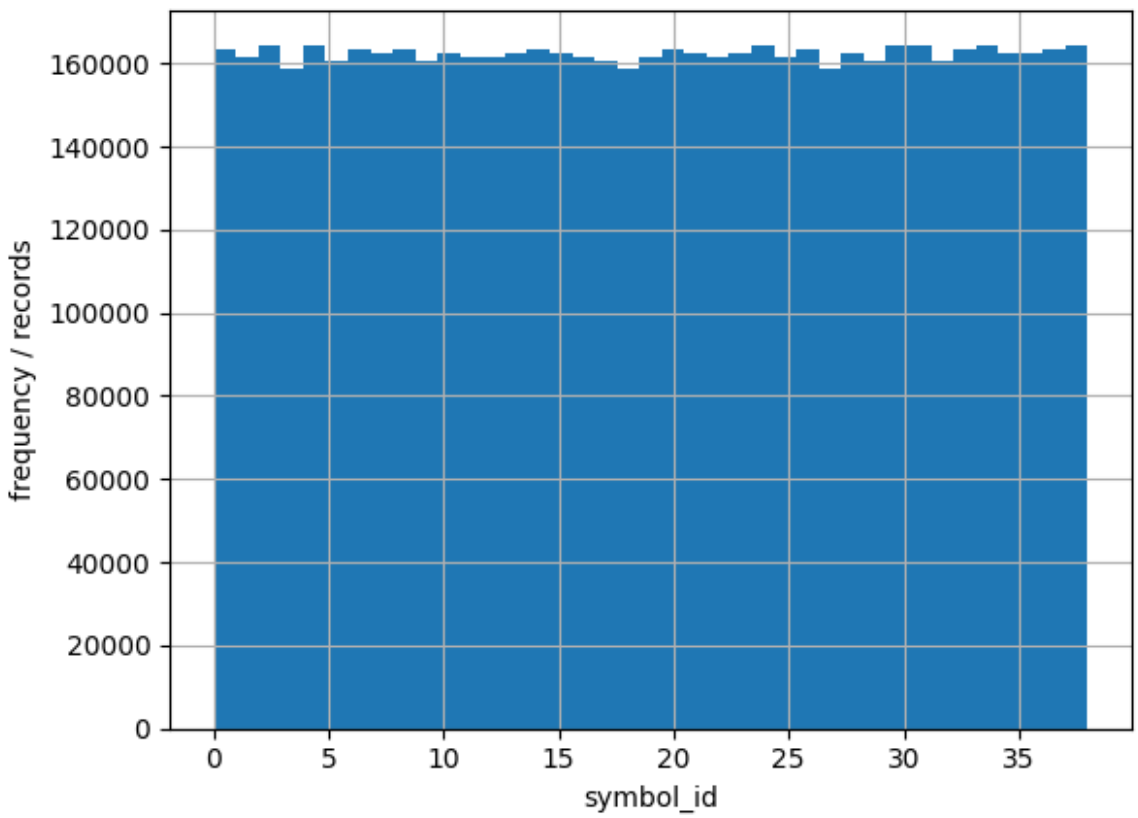

```
> train.parquet/partition_id=5/part-0.parquet
symbol_id:  0 - 38
```



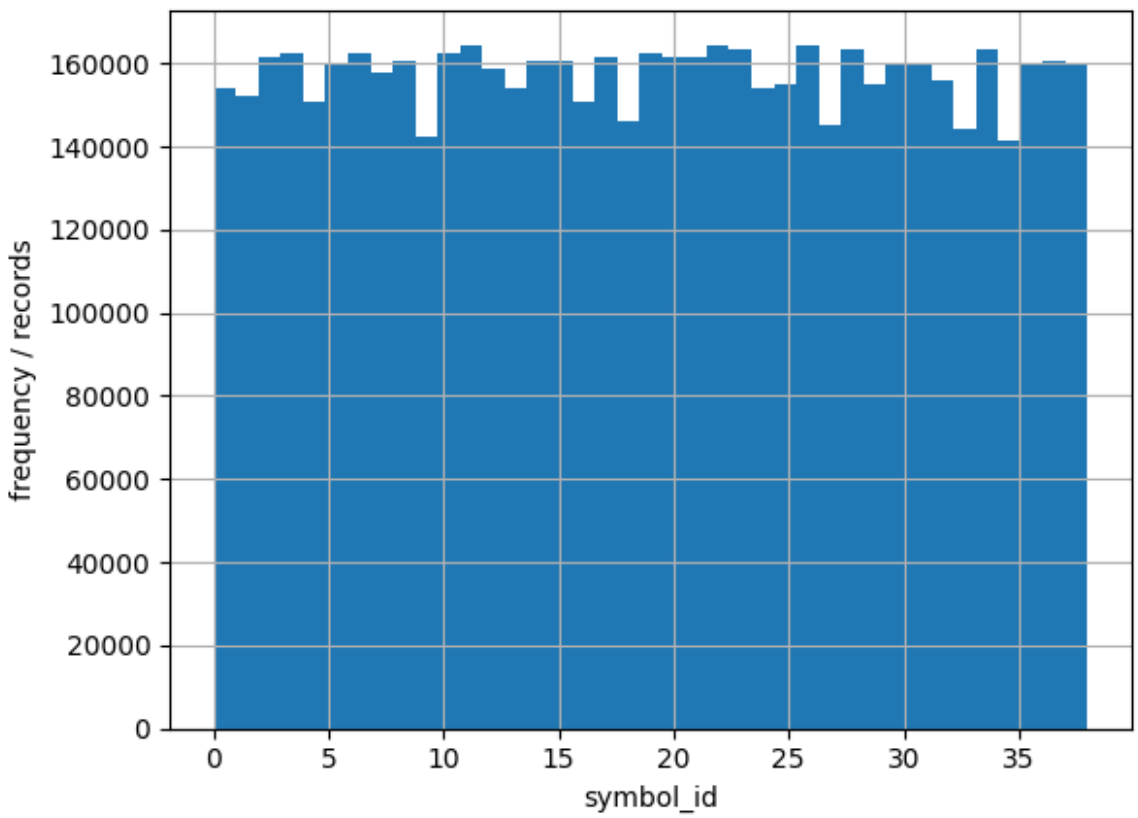

```
> train.parquet/partition_id=6/part-0.parquet
symbol_id:  0 - 38
```



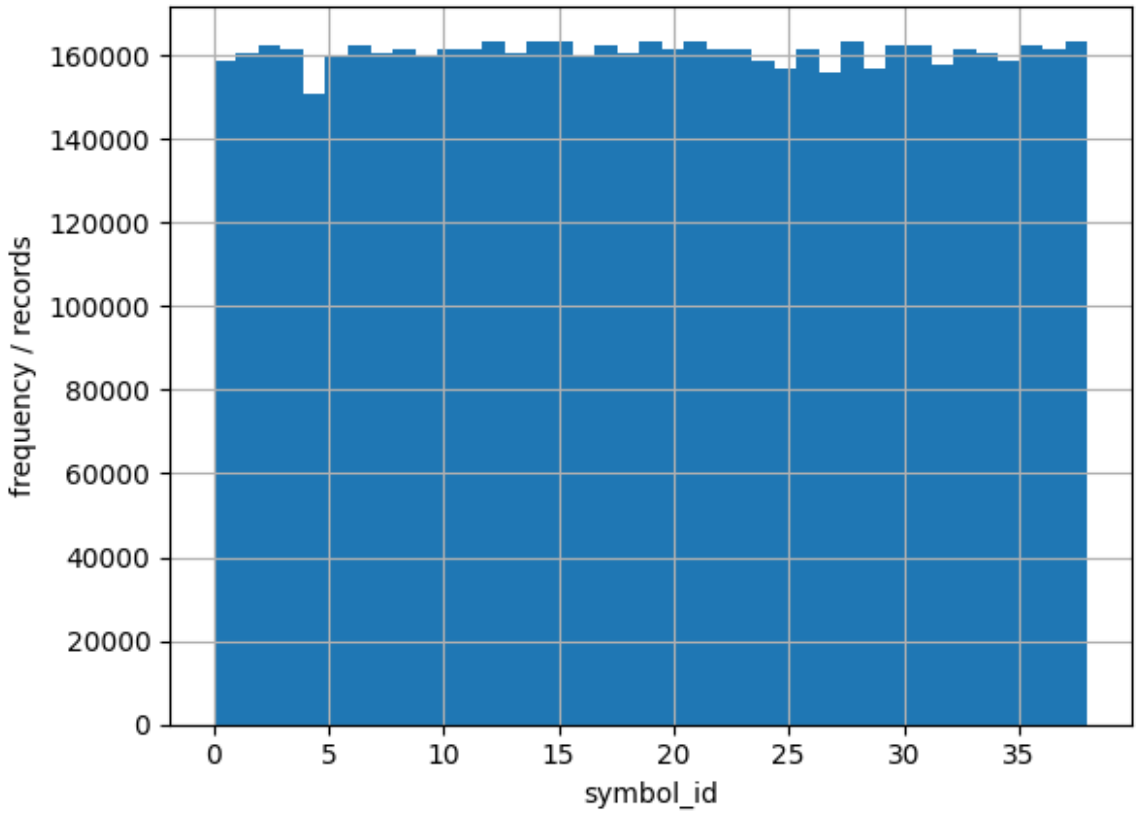

```
> train.parquet/partition_id=7/part-0.parquet
symbol_id:  0 - 38
```




```
> train.parquet/partition_id=8/part-0.parquet
symbol_id:  0 - 38
```



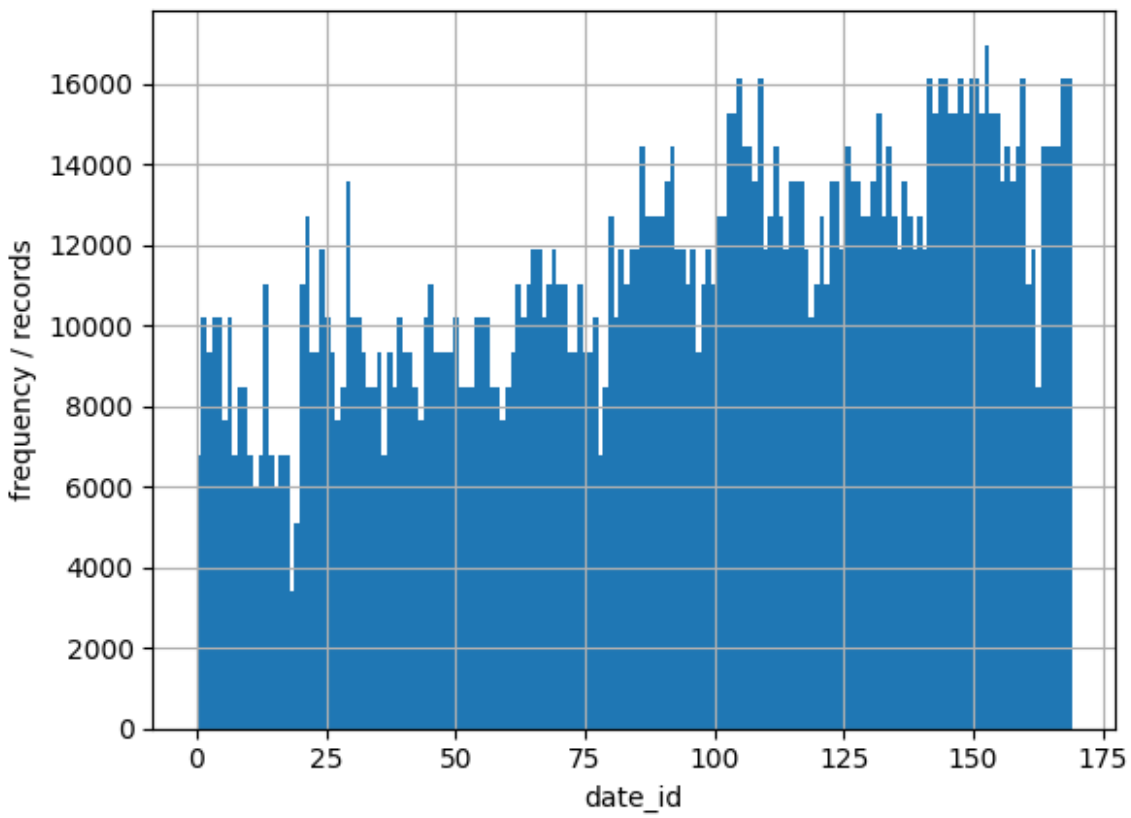

```
> train.parquet/partition_id=9/part-0.parquet
symbol_id:  0 - 38
```



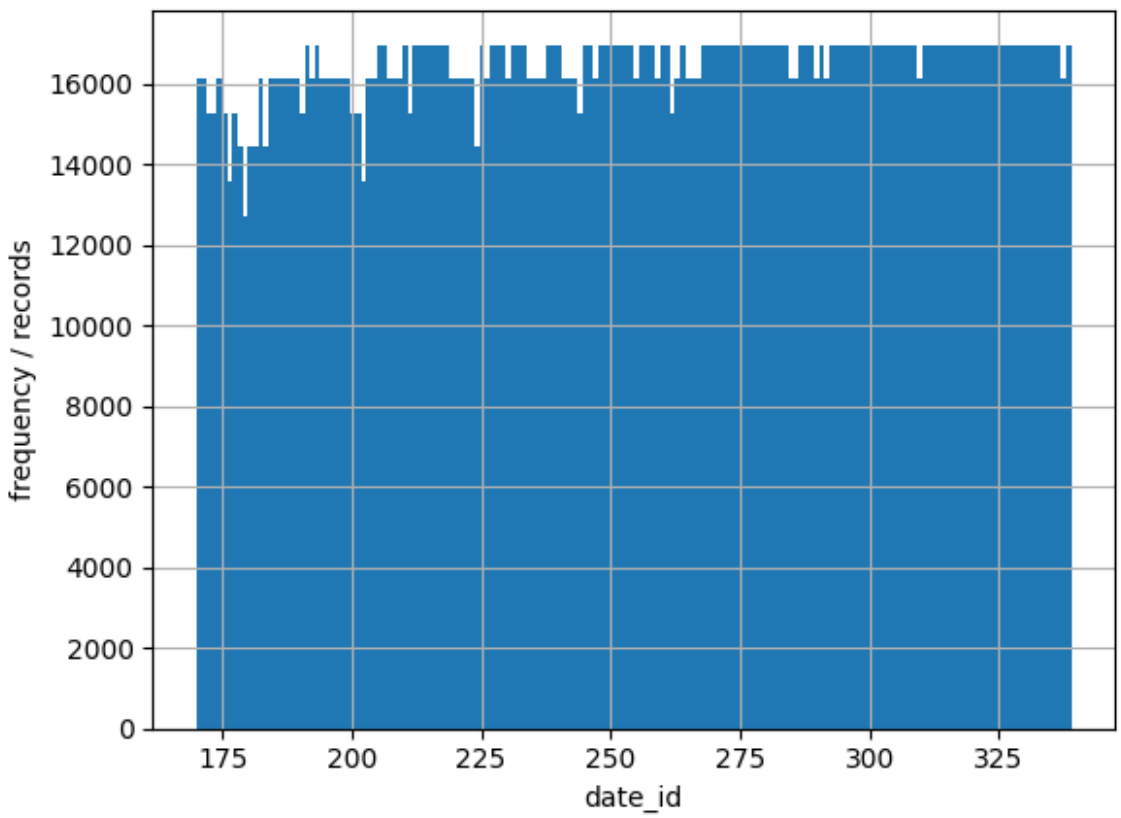

## date_id

```
for partition_id in range(10):
    print(f"> train.parquet/partition_id={partition_id}/part-
0.parquet")
    train_data =
pl.read_parquet(f"{ROOT_DIR}/train.parquet/partition_id={partition_id}
/part-0.parquet")

    print( f"date_id: ", train_data["date_id"].min(), "-",
train_data["date_id"].max())
    bins = train_data["date_id"].max() - train_data["date_id"].min() +
1
    plt.hist(train_data["date_id"], bins=bins)
    plt.xlabel("date_id")
    plt.ylabel("frequency / records")
    plt.grid()
    plt.show()
```

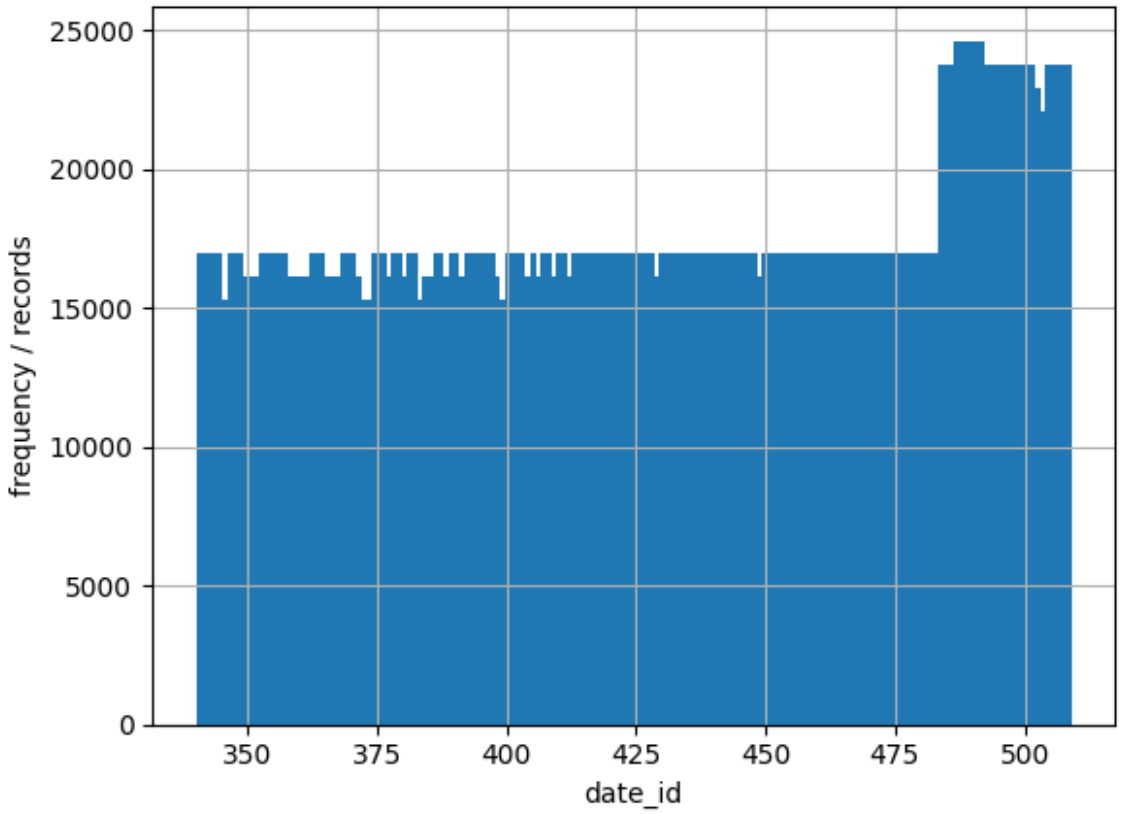
```
> train.parquet/partition_id=0/part-0.parquet
date_id:  0 - 169
```



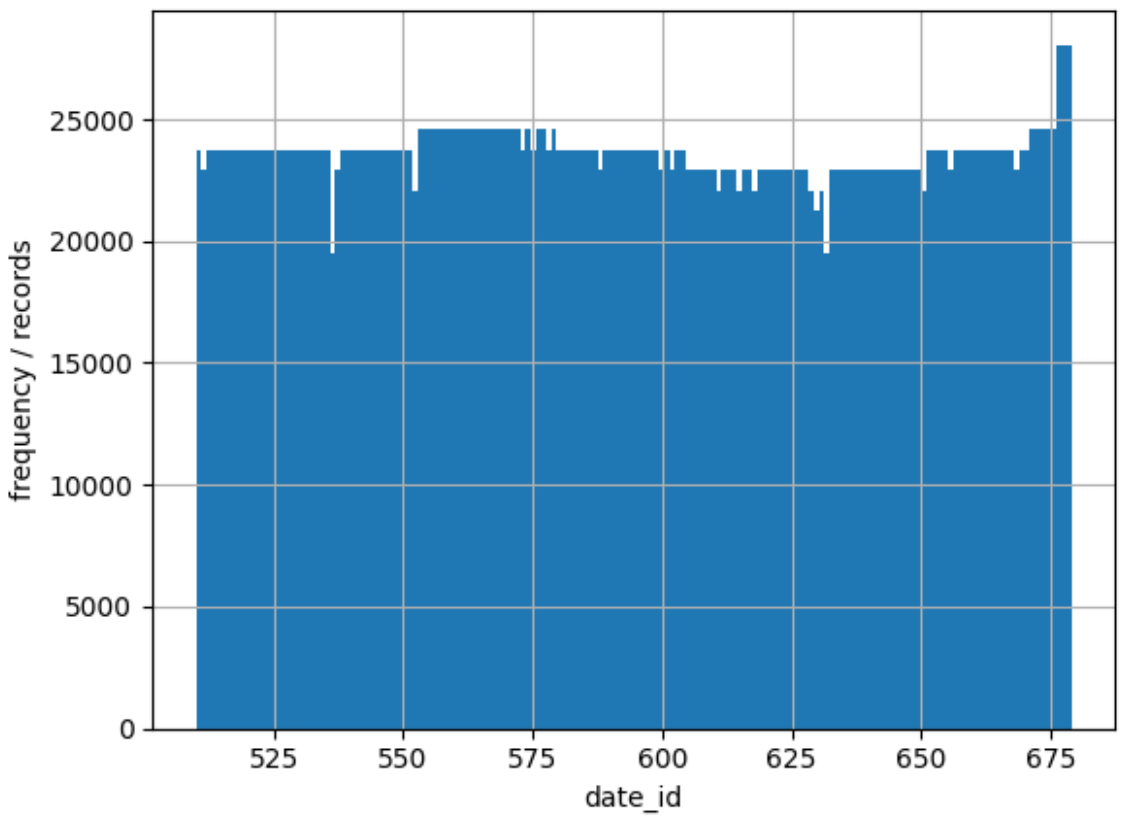

```
> train.parquet/partition_id=1/part-0.parquet
date_id:  170 - 339
```



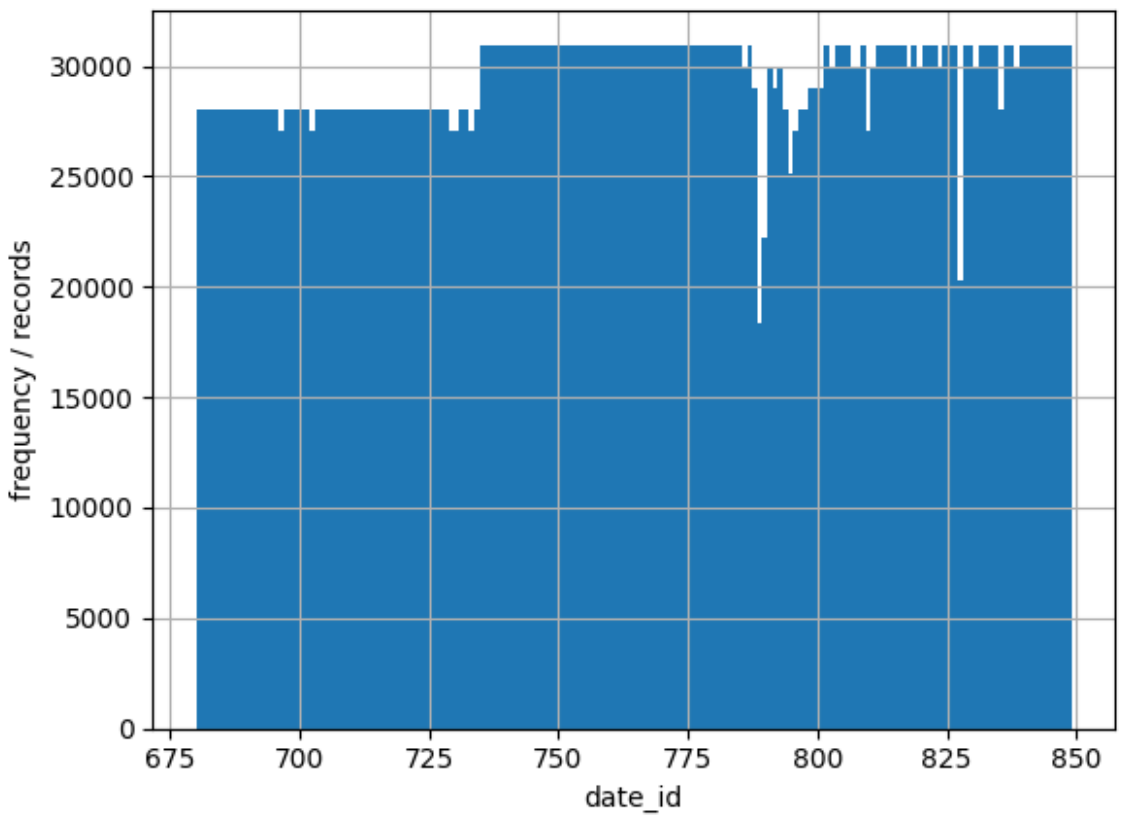

```
> train.parquet/partition_id=2/part-0.parquet
date_id:  340 - 509
```



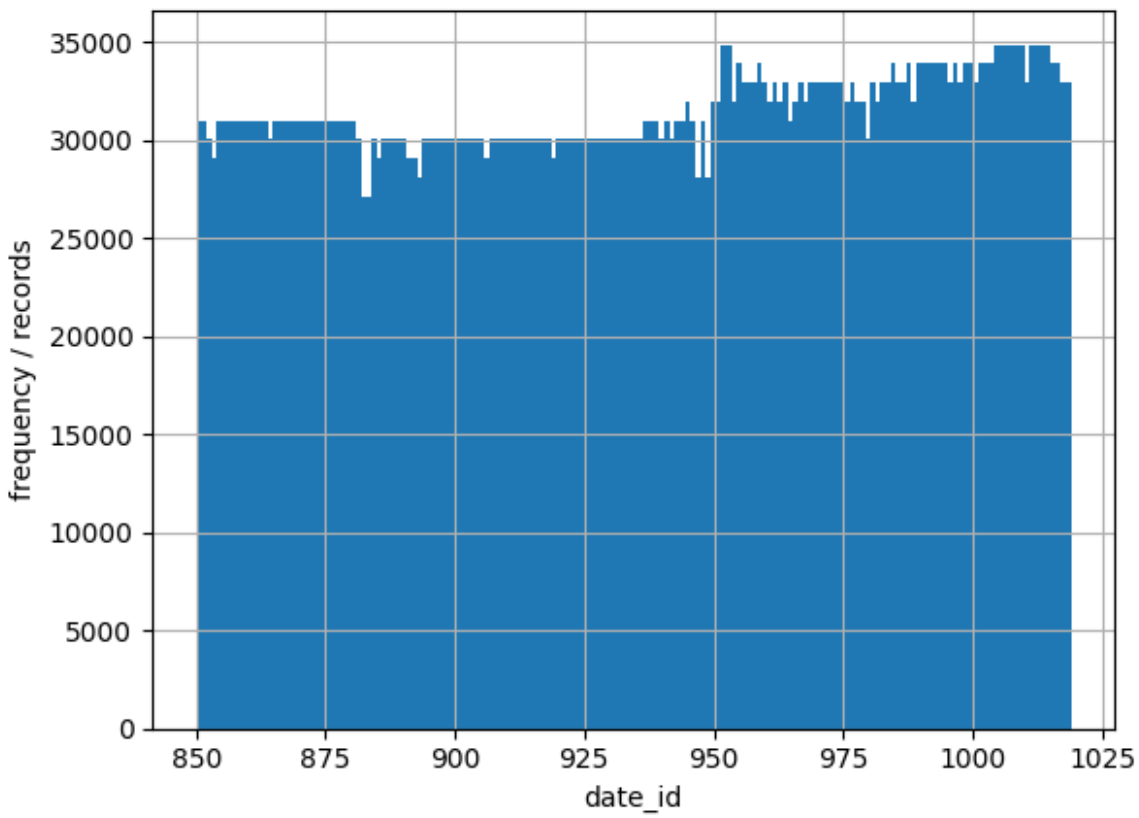

```
> train.parquet/partition_id=3/part-0.parquet
date_id:  510 - 679
```



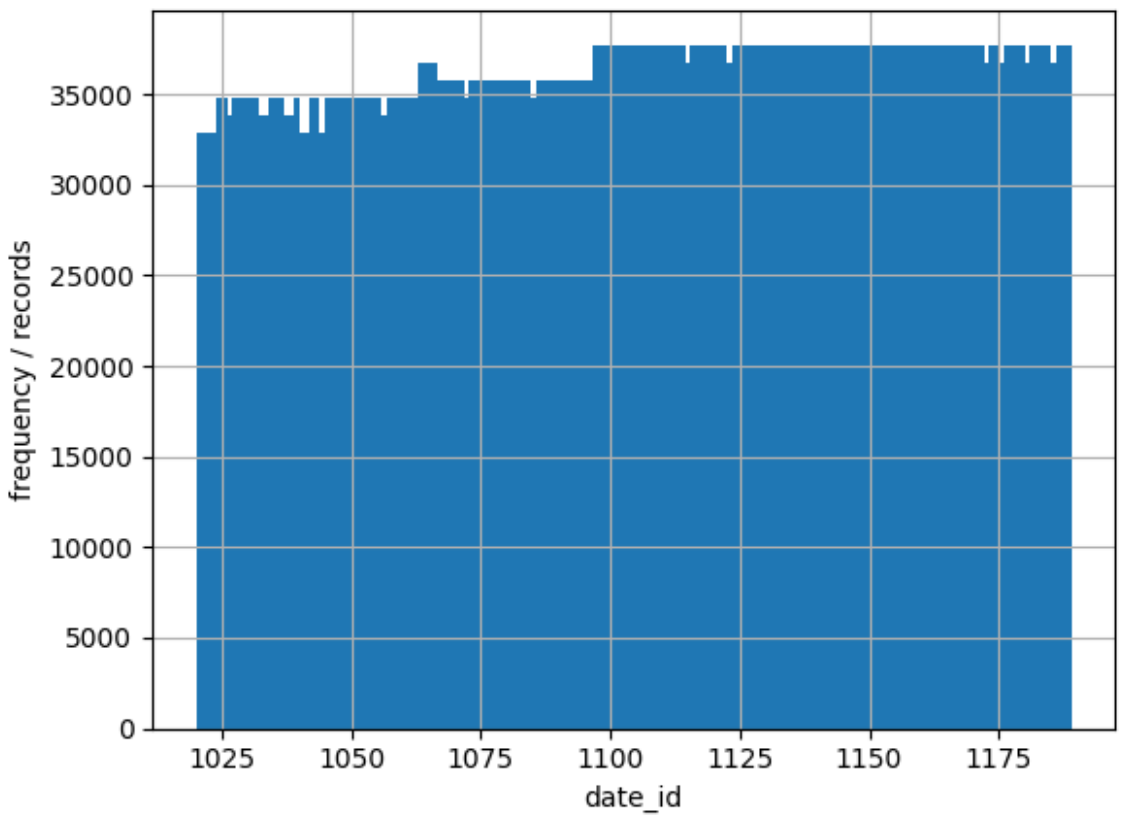

```
> train.parquet/partition_id=4/part-0.parquet
date_id:  680 - 849
```



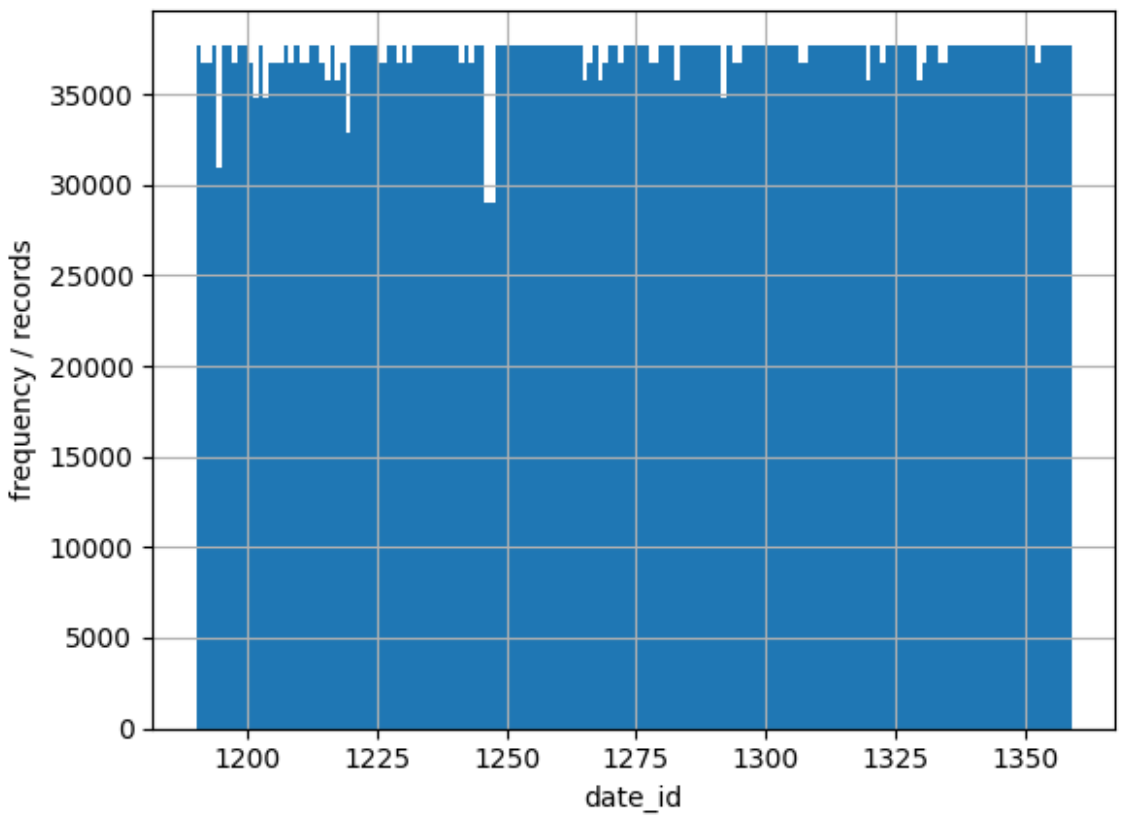

```
> train.parquet/partition_id=5/part-0.parquet
date_id:  850 - 1019
```



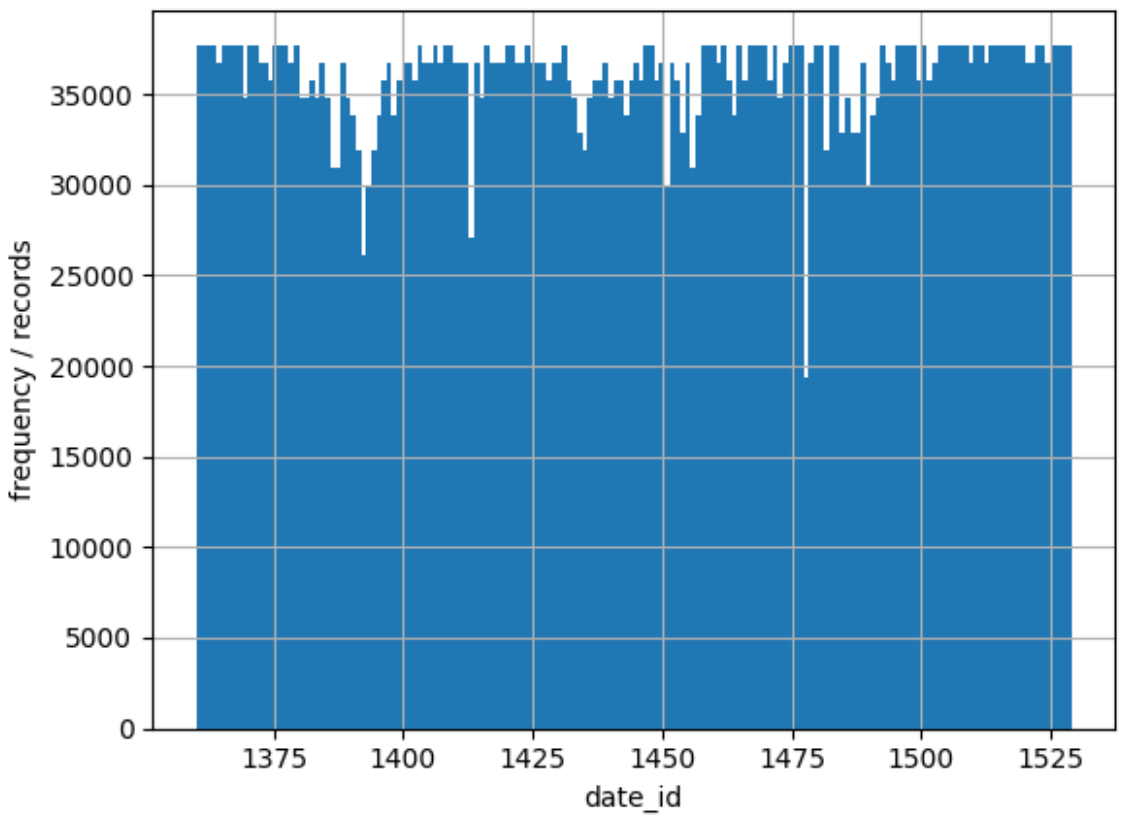

```
> train.parquet/partition_id=6/part-0.parquet
date_id:  1020 - 1189
```



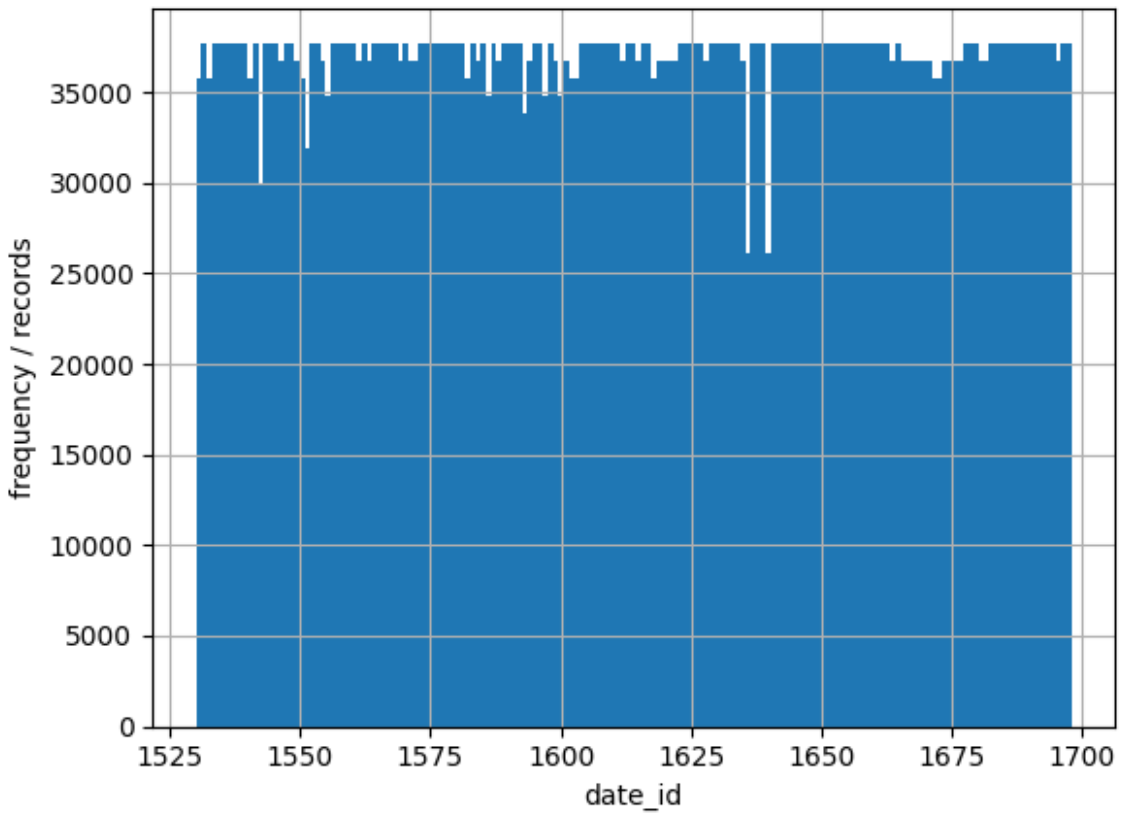

```
> train.parquet/partition_id=7/part-0.parquet
date_id:  1190 - 1359
```




```
> train.parquet/partition_id=8/part-0.parquet
date_id:  1360 - 1529
```




```
> train.parquet/partition_id=9/part-0.parquet
date_id:  1530 - 1698
```




# Test.parquet

- **test.parquet** - A mock test set which represents the structure of the unseen test set. This example set demonstrates a single batch served by the evaluation API, that is, data from a single `date_id, time_id` pair. The test set contains columns including `date_id`, `time_id`, `symbol_id`, `weight` and `feature_{00...78}`. You will not be directly using the test set or sample submission in this competition, as the evaluation API will get/set the test set and predictions.

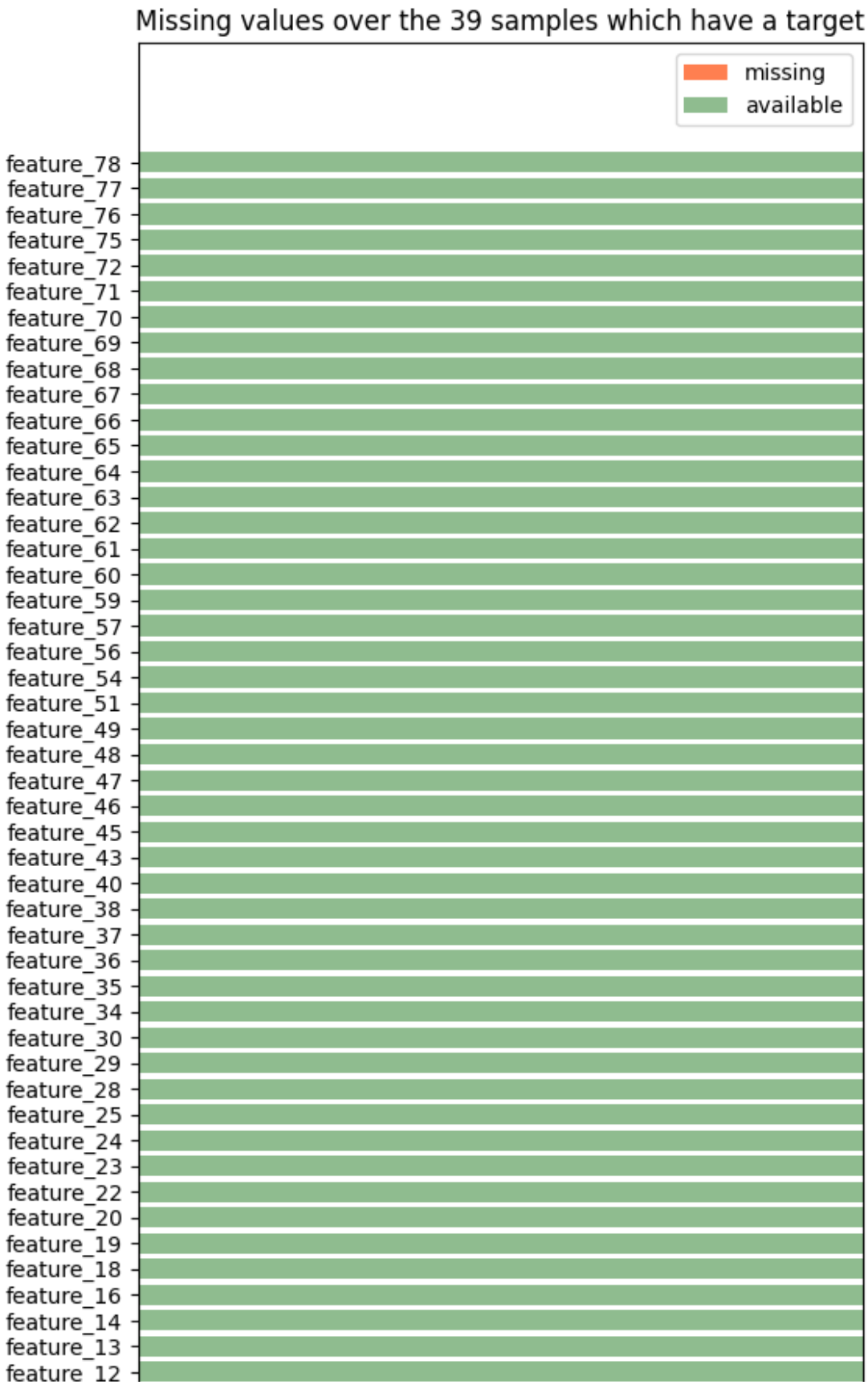
```
!tree {ROOT_DIR}/test.parquet/

/kaggle/input/jane-street-real-time-market-data-forecasting/
test.parquet/
`-- date_id=0
    `-- part-0.parquet

1 directory, 1 file

test = (
    pl.read_parquet(f"{ROOT_DIR}/test.parquet/date_id=0/part-
0.parquet")
)
test.shape
```

```
(39, 85)

test

shape: (39, 85)
┌────────┬─────────┬─────────┬────────────┬───┬──────────────┬──────────────┬──────────────┬──────────────┐
│ row_id ┆ date_id ┆ time_id ┆ symbol_id ┆ … ┆ feature_75 ┆ feature_76 ┆ feature_77 ┆ feature_78 │
│ ---    ┆ ---     ┆ ---     ┆ ---        ┆   ┆ ---          ┆ ---          ┆ ---          ┆ ---          │
│ i64    ┆ i16     ┆ i16     ┆ i8         ┆   ┆ f32          ┆ f32          ┆ f32          ┆ f32          │
╞════════╪═════════╪═════════╪════════════╪═══╪══════════════╪══════════════╪══════════════╪══════════════╡
│ 0      ┆ 0       ┆ 0       ┆ 0          ┆ … ┆ 0.0          ┆ 0.0          ┆ -0.0         ┆ -0.0         │
│ 1      ┆ 0       ┆ 0       ┆ 1          ┆ … ┆ 0.0          ┆ 0.0          ┆ 0.0          ┆ 0.0          │
│ 2      ┆ 0       ┆ 0       ┆ 2          ┆ … ┆ 0.0          ┆ 0.0          ┆ -0.0         ┆ -0.0         │
│ 3      ┆ 0       ┆ 0       ┆ 3          ┆ … ┆ 0.0          ┆ 0.0          ┆ -0.0         ┆ -0.0         │
│ 4      ┆ 0       ┆ 0       ┆ 4          ┆ … ┆ 0.0          ┆ 0.0          ┆ 0.0          ┆ 0.0          │
│ …      ┆ …       ┆ …       ┆ …          ┆ … ┆ …            ┆ …            ┆ …            ┆ …            │
│ 34     ┆ 0       ┆ 0       ┆ 34         ┆ … ┆ 0.0          ┆ 0.0          ┆ 0.0          ┆ 0.0          │
│ 35     ┆ 0       ┆ 0       ┆ 35         ┆ … ┆ 0.0          ┆ 0.0          ┆ -0.0         ┆ -0.0         │
│ 36     ┆ 0       ┆ 0       ┆ 36         ┆ … ┆ 0.0          ┆ 0.0          ┆ 0.0          ┆ 0.0          │
│ 37     ┆ 0       ┆ 0       ┆ 37         ┆ … ┆ 0.0          ┆ 0.0          ┆ 0.0          ┆ 0.0          │
│ 38     ┆ 0       ┆ 0       ┆ 38         ┆ … ┆ 0.0          ┆ 0.0          ┆ -0.0         ┆ -0.0         │
└────────┴─────────┴─────────┴────────────┴───┴──────────────┴──────────────┴──────────────┴──────────────┘
```

## Missing values

```
supervised_usable = (
    test
)

missing_count = (
    supervised_usable
    .null_count()
    .transpose(include_header=True,
```

```
                header_name='feature',
                column_names=['null_count'])
    .sort('null_count', descending=True)
    .with_columns((pl.col('null_count') /
len(supervised_usable)).alias('null_ratio'))
)

plt.figure(figsize=(6, 20))
plt.title(f'Missing values over the {len(supervised_usable)} samples
which have a target')
plt.barh(np.arange(len(missing_count)),
missing_count.get_column('null_ratio'), color='coral',
label='missing')
plt.barh(np.arange(len(missing_count)),
         1 - missing_count.get_column('null_ratio'),
         left=missing_count.get_column('null_ratio'),
         color='darkseagreen', label='available')
plt.yticks(np.arange(len(missing_count)),
missing_count.get_column('feature'))
plt.gca().xaxis.set_major_formatter(PercentFormatter(xmax=1,
decimals=0))
plt.xlim(0, 1)
plt.legend()
plt.show()
```


Missing values over the 39 samples which have a target
missing
available
feature_78
feature_77
feature_76
feature_75
feature_72
feature_71
feature_70
feature_69
feature_68
feature_67
feature_66
feature_65
feature_64
feature_63
feature_62
feature_61
feature_60
feature_59
feature_57
feature_56
feature_54
feature_51
feature_49
feature_48
feature_47
feature_46
feature_45
feature_43
feature_40
feature_38
feature_37
feature_36
feature_35
feature_34
feature_30
feature_29
feature_28
feature_25
feature_24
feature_23
feature_22
feature_20
feature_19
feature_18
feature_16
feature_14
feature_13
feature_12

# lags.parquet

- `lags.parquet` - Values of `responder_{0...8}` lagged by one `date_id`. The evaluation API serves the entirety of the lagged responders for a `date_id` on that date_id's first `time_id`. In other words, all of the previous date's responders will be served at the first time step of the succeeding date.

```
!tree {ROOT_DIR}/lags.parquet
```

```
/kaggle/input/jane-street-real-time-market-data-forecasting/
lags.parquet
`-- date_id=0
    `-- part-0.parquet

1 directory, 1 file
```

```
lags = (
    pl.read_parquet(f"{ROOT_DIR}/lags.parquet/date_id=0/part-
0.parquet")
)
lags.shape
```

```
(39, 12)
```

```
lags.columns
```

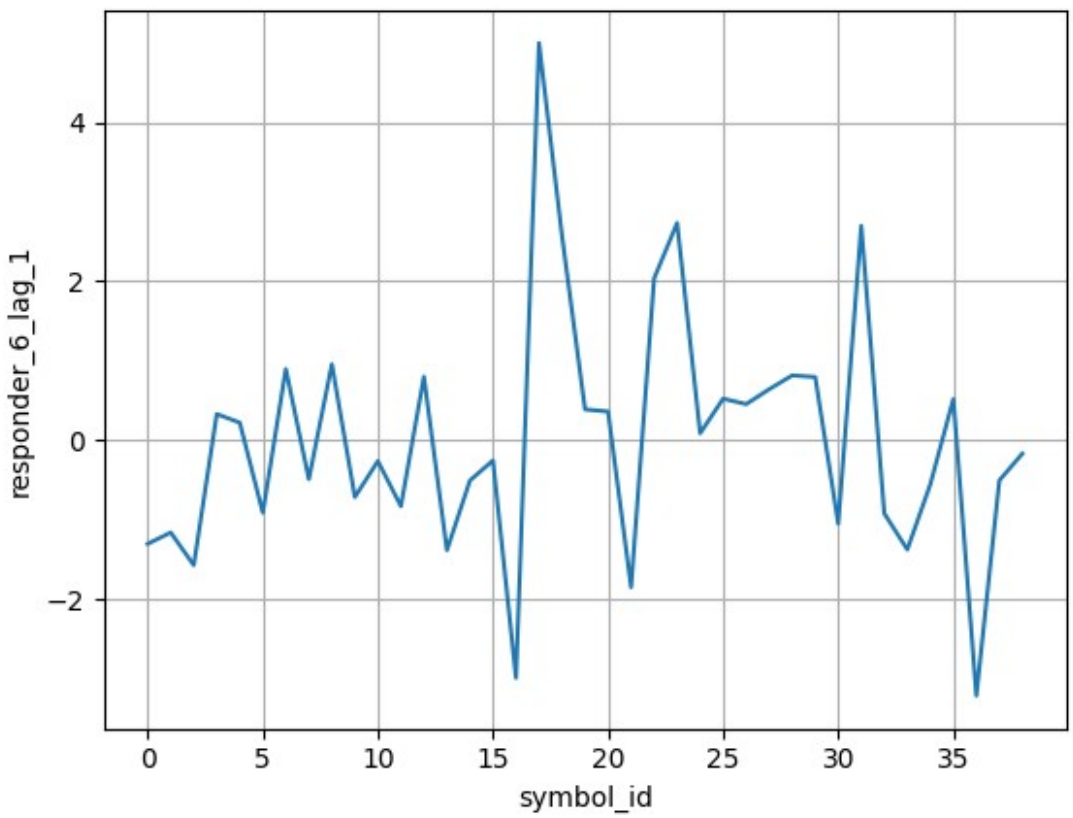
```
['date_id',
 'time_id',
 'symbol_id',
 'responder_0_lag_1',
 'responder_1_lag_1',
 'responder_2_lag_1',
 'responder_3_lag_1',
 'responder_4_lag_1',
 'responder_5_lag_1',
 'responder_6_lag_1',
 'responder_7_lag_1',
 'responder_8_lag_1']
```

```
lags
```

```
shape: (39, 12)
┌─────────┬─────────┬───────────┬─────────────┬───┬─────────────┬─────────
──────┬─────────────┐
│ date_id ┆ time_id ┆ symbol_id ┆ responder_ ┆ … ┆ responder_ ┆
responder_ ┆ responder ┆ responder │
│ ---     ┆ ---     ┆ ---       ┆ 0_lag_1    ┆   ┆ 5_lag_1    ┆
6_lag_1    ┆ _7_lag_1  ┆ _8_lag_1  │
│ i16     ┆ i16     ┆ i8        ┆ ---        ┆   ┆ ---        ┆ ---
┆ ---       ┆ ---       │
│         ┆         ┆           ┆ f32        ┆   ┆ f32        ┆ f32
```

```
┆ f32       ┆ f32       │
╞═══════════╪═══════════╪═══════════╪═══════════╪═══╪═══════════╪═══════
═══╪═══════════╪═══════════╡
│ 0         ┆ 0         ┆ 0         ┆ -0.442215 ┆ … ┆ -0.036595 ┆ -
1.305746  ┆ -0.795677 ┆ -0.143724 │
│ 0         ┆ 0         ┆ 1         ┆ -0.651829 ┆ … ┆ -0.615652 ┆ -
1.162801  ┆ -1.205924 ┆ -1.245934 │
│ 0         ┆ 0         ┆ 2         ┆ -0.656373 ┆ … ┆ -0.378265 ┆ -
1.57429   ┆ -1.863071 ┆ -0.027343 │
│ 0         ┆ 0         ┆ 3         ┆ -0.188186 ┆ … ┆ -0.054984 ┆
0.329152  ┆ -0.965471 ┆ 0.576635  │
│ 0         ┆ 0         ┆ 4         ┆ -0.257462 ┆ … ┆ -0.597093 ┆
0.219856  ┆ -0.276356 ┆ -0.90479  │
│ …         ┆ …         ┆ …         ┆ …         ┆ … ┆ …         ┆ …
┆ …         ┆ …         │
│ 0         ┆ 0         ┆ 34        ┆ -0.185392 ┆ … ┆ -0.443875 ┆ -
0.556474  ┆ -1.122211 ┆ -0.884185 │
│ 0         ┆ 0         ┆ 35        ┆ -0.308923 ┆ … ┆ 0.424937  ┆
0.518839  ┆ -0.687369 ┆ 1.440577  │
│ 0         ┆ 0         ┆ 36        ┆ -0.074661 ┆ … ┆ -1.601274 ┆ -
3.216254  ┆ -1.249338 ┆ -2.868875 │
│ 0         ┆ 0         ┆ 37        ┆ -0.658366 ┆ … ┆ -1.562932 ┆ -
0.506418  ┆ -1.355508 ┆ -2.630985 │
│ 0         ┆ 0         ┆ 38        ┆ 0.572666  ┆ … ┆ -0.501347 ┆ -
0.169114  ┆ 0.457801  ┆ -0.136777 │
└───────────┴───────────┴───────────┴───────────┴───┴───────────┴───────
───┴───────────┴───────────┘
```

```
plt.plot(lags["responder_6_lag_1"])
plt.grid()
plt.xlabel("symbol_id")
plt.ylabel("responder_6_lag_1")
plt.show()
```




## Missing value (Null) analysis

```
train_all = pl.scan_parquet("/kaggle/input/jane-street-real-time-
market-data-forecasting/train.parquet")

# Count null(NaN) for eatch columns (group by date_id)
null_count_per_date_id =
train_all.group_by("date_id").agg(pl.all().null_count()).collect()
null_count_per_date_id
```

```
shape: (1_699, 93)
┌─────────┬─────────┬────────────┬─────────┬───┬─────────────┬─────────────
──┬─────────────┬─────────────┐
│ date_id ┆ time_id ┆ symbol_id ┆ weight ┆ … ┆ responder_6 ┆
responder_7 ┆ responder_ ┆ partition_ │
│ ---     ┆ ---     ┆ ---        ┆ ---    ┆   ┆ ---         ┆ ---
┆ 8          ┆ id          │
│ i16     ┆ u32     ┆ u32        ┆ u32    ┆   ┆ u32         ┆ u32
┆ ---        ┆ ---         │
│         ┆         ┆            ┆        ┆   ┆             ┆
┆ u32        ┆ u32         │
```

```
╞═══════╪═══════╪═══════╪═══════╪═══╪═══════╪═══════
═══╪═══════╪═══════╡
│ 486   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 810   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 715   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 215   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 146   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ …     ┆ …     ┆ …     ┆ …     ┆ … ┆ …     ┆ …
┆ …     ┆ …     │
│ 1122  ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 1226  ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 1134  ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 1089  ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
│ 598   ┆ 0     ┆ 0     ┆ 0     ┆ … ┆ 0     ┆ 0
┆ 0     ┆ 0     │
└───────┴───────┴───────┴───────┴───┴───────┴───────
───┴───────┴───────┘
```

```
# Counter number of records group by date_id
records_date_id =
train_all.group_by("date_id").agg(pl.count().alias("num_records")).col
lect()
records_date_id
```

```
/tmp/ipykernel_17/1203909370.py:2: DeprecationWarning: `pl.count()` is
deprecated. Please use `pl.len()` instead.
  records_date_id =
train_all.group_by("date_id").agg(pl.count().alias("num_records")).col
lect()
```

```
shape: (1_699, 2)
```

| date_id | num_records |
|---|---|
| i16 | u32 |
| 420 | 16980 |
| 864 | 30008 |
| 1632 | 37752 |
| 84 | 11886 |
| 539 | 23772 |

```
┌──────────┬──────────────┐
│ …        ┆ …            │
│ 604      ┆ 23772        │
│ 702      ┆ 27104        │
│ 1646     ┆ 37752        │
│ 821      ┆ 30976        │
│ 464      ┆ 16980        │
└──────────┴──────────────┘
```

```
# Count null(NaN) for all features columns (group by date_id)
features = [f"feature_{i:02d}" for i in range(79) ]
sum_null_count_per_date_id = null_count_per_date_id.with_columns(
    null_count=pl.sum_horizontal(features)
).select(
    "date_id", "null_count"
).join( records_date_id, on="date_id", how="inner").to_pandas()
sum_null_count_per_date_id["null_ratio"] =
sum_null_count_per_date_id["null_count"] /
sum_null_count_per_date_id["num_records"] / 79
sum_null_count_per_date_id
```

```
      date_id  null_count  num_records  null_ratio
0         420       83014        16980    0.061885
1         864       12983        30008    0.005477
2        1632       20396        37752    0.006839
3          84      121174        11886    0.129047
4         539       20168        23772    0.010739
...       ...         ...          ...         ...
1694      604       20177        23772    0.010744
1695      702       11796        27104    0.005509
1696     1646       20330        37752    0.006817
1697      821       14356        30976    0.005867
1698      464       82368        16980    0.061404

[1699 rows x 4 columns]
```