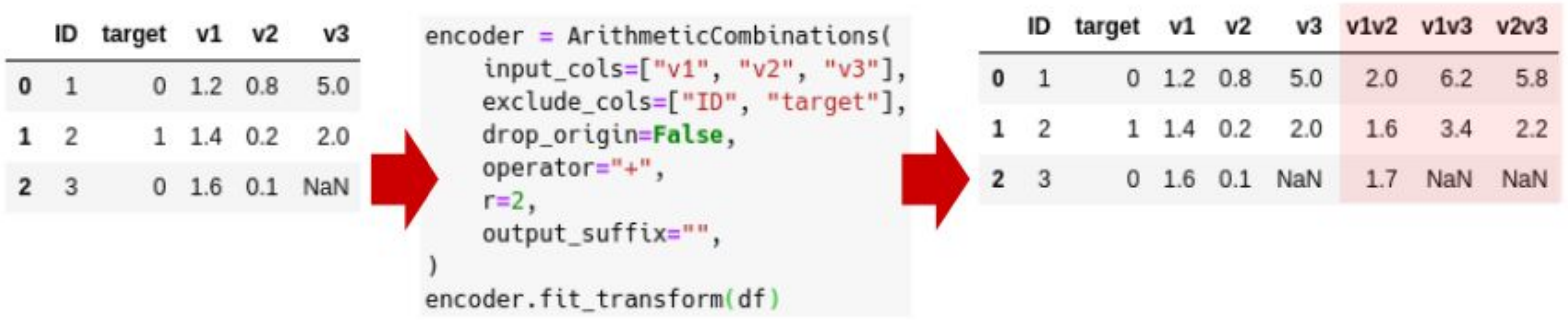
xfeat: Feature Engineering and Exploration Library

Kohei Ozaki at Preferred Networks

Main Idea: DataFrame-IN, DataFrame-OUT

- xfeat provides feature encoders and selectors.
- DataFrame as Input. DataFrame as Output.
- Support both **pandas** and **cuDF** dataframe.



	ID	target	v1	v2	v3
0	1	0	1.2	0.8	5.0
1	2	1	1.4	0.2	2.0
2	3	0	1.6	0.1	NaN

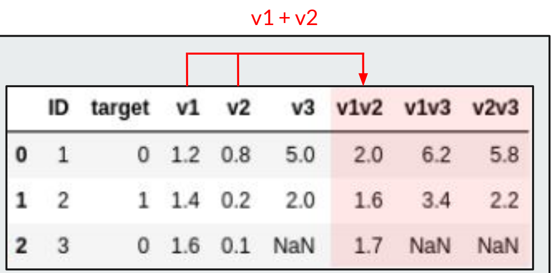
```
encoder = ArithmeticCombinations(
    input_cols=["v1", "v2", "v3"],
    exclude_cols=["ID", "target"],
    drop_origin=False,
    operator="+",
    r=2,
    output_suffix="",
)
encoder.fit_transform(df)
```

	ID	target	v1	v2	v3	v1v2	v1v3	v2v3
0	1	0	1.2	0.8	5.0	2.0	6.2	5.8
1	2	1	1.4	0.2	2.0	1.6	3.4	2.2
2	3	0	1.6	0.1	NaN	1.7	NaN	NaN

Main features: Encoders and Selectors

- Numerical:
  - SelectNumerical
  - ArithmeticCombinations
- Categorical:
  - SelectCategorical
  - LabelEncoder
  - Target Encoder
  - ConcatCombination
  - CountEncoder
  - UserDefinedLabelEncoder
- Feature Selection (Selectors): GBDTFeatureSelector
- Feature Exploration (Selectors): GBDTFeatureExploration

Generate features by arithmetic combinations of numerical columns.



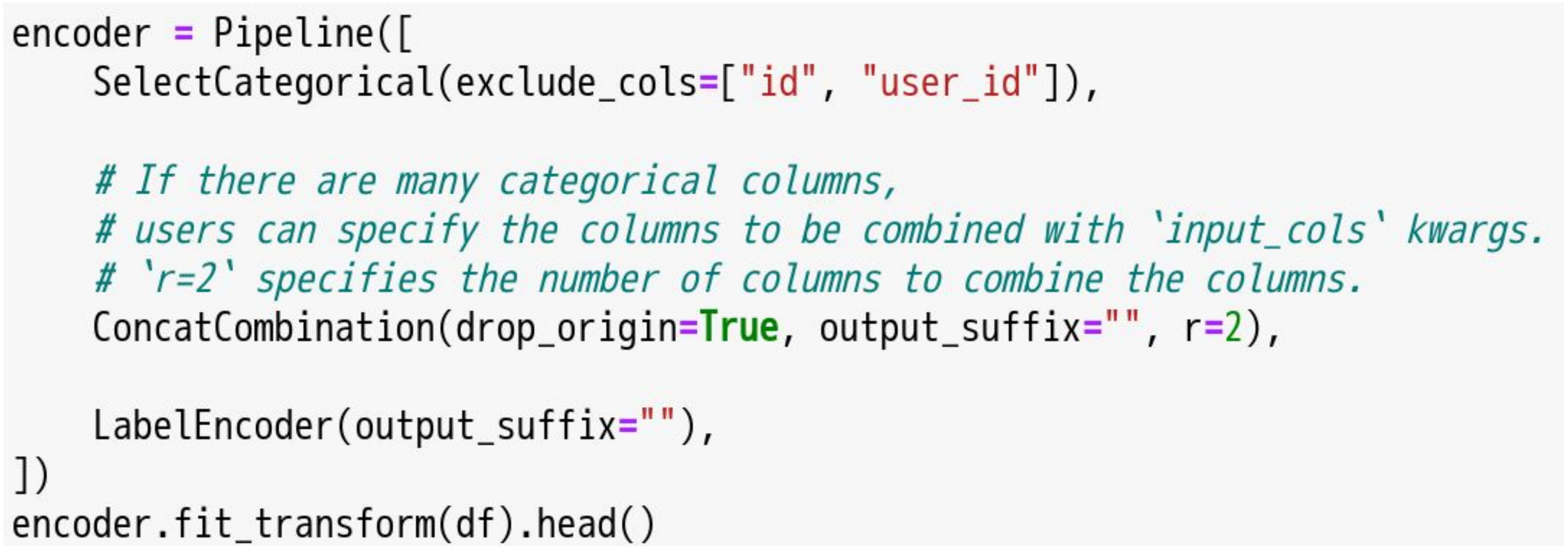
	ID	target	v1	v2	v3	v1v2	v1v3	v2v3
0	1	0	1.2	0.8	5.0	2.0	6.2	5.8
1	2	1	1.4	0.2	2.0	1.6	3.4	2.2
2	3	0	1.6	0.1	NaN	1.7	NaN	NaN

Generate features by concatenating string values of categoricals.

Filter-based method using LightGBM. Its hyperparameter is tuned by Optuna.

Main features: Pipeline

Combine multiple encoders into a single encoder object.



```
encoder = Pipeline([
    SelectCategorical(exclude_cols=["id", "user_id"]),

    # If there are many categorical columns,
    # users can specify the columns to be combined with `input_cols` kwargs.
    # `r=2` specifies the number of columns to combine the columns.
    ConcatCombination(drop_origin=True, output_suffix="", r=2),

    LabelEncoder(output_suffix=""),
])
encoder.fit_transform(df).head()
```

Main features: Serialize/Deserialize

Serialize/Deserialize Encoders with pickle.

This makes it easier to separate training step and inference step.

1. Fit parameters on a train set.

```
encoder = Pipeline([
    SelectCategorical(exclude_cols=["id", "user_id"]),
    LabelEncoder(output_suffix=""),
])
df_train_encoded = encoder.fit_transform(df_train)

with open("label_encoder.pkl", "wb") as f:
    pickle.dump(encoder, f)
```

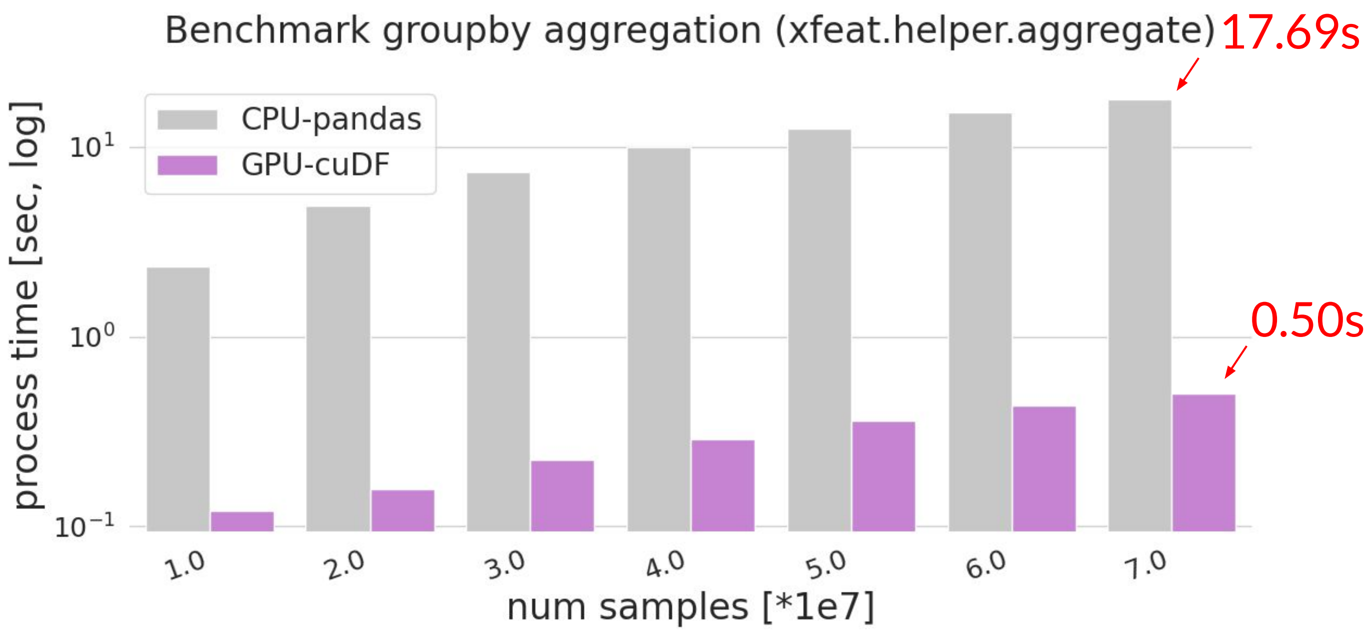
2. Transform on a test set later.

```
with open("label_encoder.pkl", "rb") as f:
    encoder = pickle.load(f)

encoder.transform(df_test).head()
```

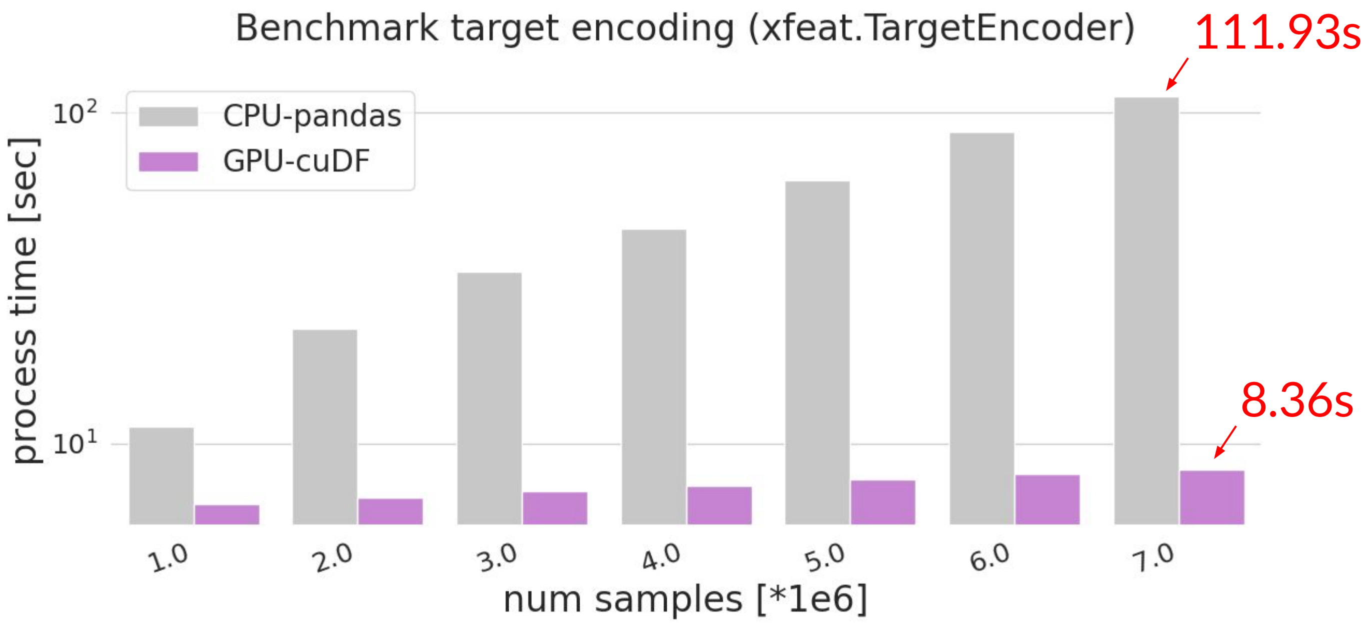
Benchmark: Group-by aggregation

Encoders can be greatly accelerated with cuDF and CuPy.



Benchmark: Target Encoding

In Target Encoding, aggregation is computed on the GPUs using CuPy.

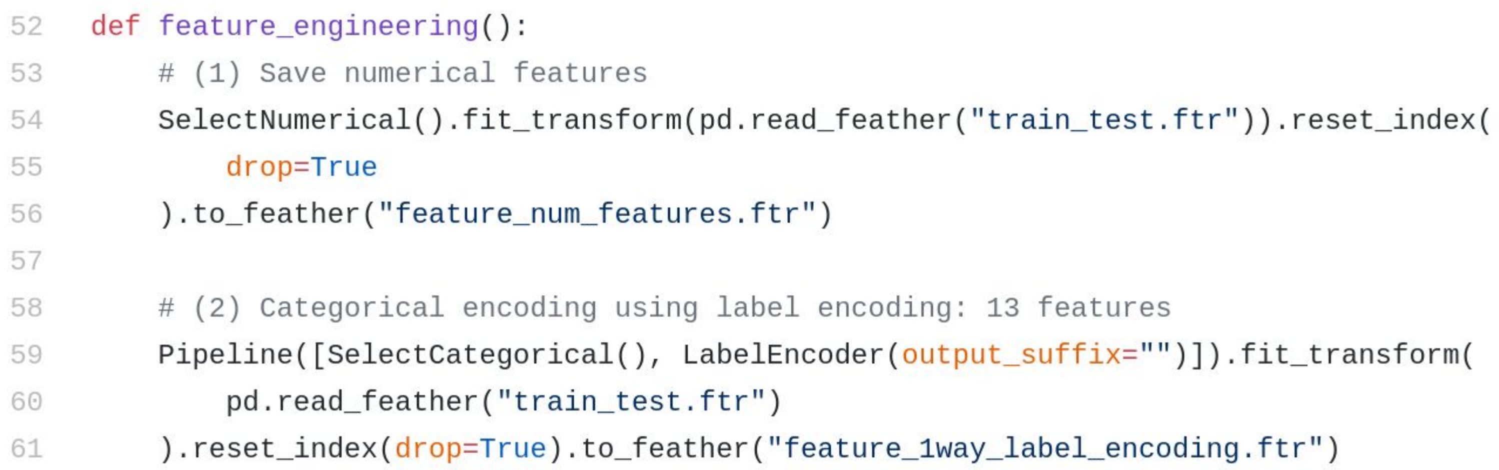


Practical result on Kaggle competition

With using xfeat, I got 12th place LB score with less than 200 lines.
I plan to upload the solution code on kaggle.com after releasing xfeat.

The initial OSS release is planned on June.

Appendix: solution code (1 of 3)



```python
def feature_engineering():
    # (1) Save numerical features
    SelectNumerical().fit_transform(pd.read_feather("train_test.ftr")).reset_index(
        drop=True
    ).to_feather("feature_num_features.ftr")

    # (2) Categorical encoding using label encoding: 13 features
    Pipeline([SelectCategorical(), LabelEncoder(output_suffix="")]).fit_transform(
        pd.read_feather("train_test.ftr")
    ).reset_index(drop=True).to_feather("feature_1way_label_encoding.ftr")
```

Appendix: solution code (2 of 3)

```python
    # (4) 3-order combination of categorical features
    # Use `include_cols=` kwargs to reduce the total count of combinations.
    # 66 features (12 * 11 / 2 = 66)
    Pipeline(
        [
            SelectCategorical(),
            ConcatCombination(drop_origin=True, include_cols=["v22"], r=3),
            LabelEncoder(output_suffix=""),
        ]
    ).fit_transform(pd.read_feather("train_test.ftr")).reset_index(
        drop=True
    ).to_feather(
        "feature_3way_including_v22_label_encoding.ftr"
    )
```

Appendix: solution code (3 of 3)

```
    # (6) 2-order Arithmetic combinations.
    Pipeline(
        [
            SelectNumerical(),
            ArithmeticCombinations(
                exclude_cols=["target"], drop_origin=True, operator="+", r=2,
            ),
        ]
    ).fit_transform(pd.read_feather("train_test.ftr")).reset_index(
        drop=True
    ).to_feather(
        "feature_arithmetic_combi2.ftr"
    )
```