New Machine Learning Pipeline Framework with New ONNX Operators

Takuya Nakaike, Mori Ohara, Hong Min, Alexandre Eichenberger

IBM Research

ONNX Roadmap Discussion on September 9, 2021

Summary

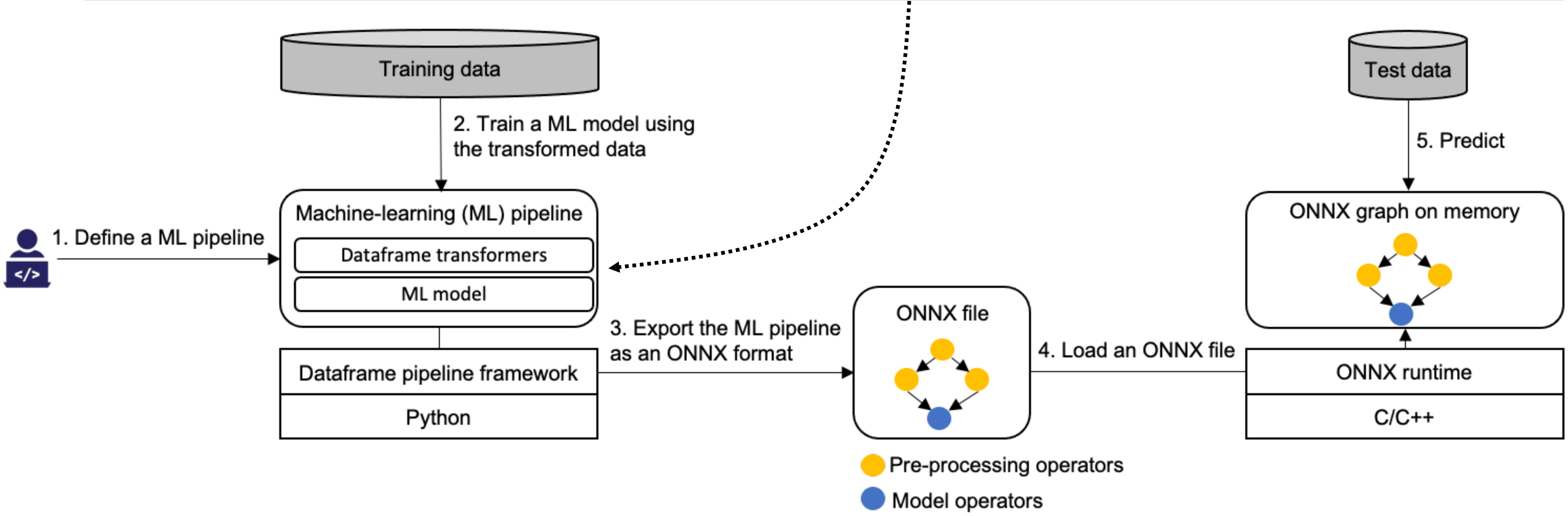
- Motivation from findings in Kaggle use cases
  - Pandas dataframe is very popular to write the data pre-processing code.
  - There is no ML pipeline framework to represent typical patterns of data preprocessing.
    - A new feature cannot be calculated from multiple features.
  - ONNX lacks a few operators to represent typical patterns of data pre-processing.

- Our proposal
  - New ML pipeline framework on Python to convert typical data-preprocessing patterns on pandas dataframe into ONNX
  - Three new ONNX operators to represent typical data-preprocessing patterns.
    - Date: Parse a date string to extract time features such as a year and a month.
    - StringConcatenator: Concatenate multiple strings
    - StringSplitter: Split a string based on a given separator or index

New ML Pipeline: Dataframe Pipeline

https://github.com/IBM/dataframe-pipeline

```
…
# Define a dataframe pipeline
pipeline = dft.DataframePipeline(steps=[
  dft.StringConcatenator(inputs=[('col_1', 'col_2')], outputs=['col_3'], separator='_')
  dft.LabelEncoder(inputs=['col_1', 'col_2', 'col_3'], outputs=['col_1', 'col_2', 'col_3']),
  dft.ColumnSelector(columns=['col_1', 'col_2', 'col_3']),
])
…
pipeline.export(xgb_onnx_model, 'pipeline.onnx')
```

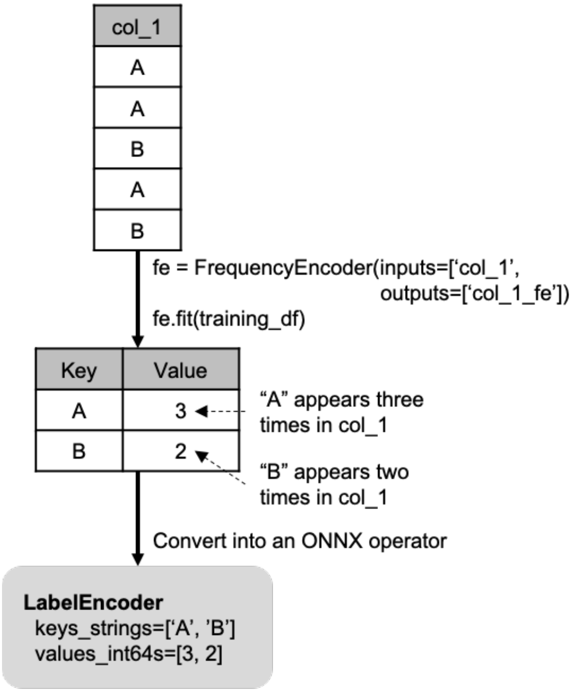


Mapping from Dataframe Trasfomers into ONNX operators

Dataframe transformer	ONNX operator
FunctionTransformer	Arithmetic operators (e.g., Add)
MapTransfomer	LabelEncoder
LabelEncoder	LabelEncoder
OneHotEncoder	OneHotEncoder
FrequencyEncoder	LabelEncoder
Aggregator	LabelEncoder
Scaler	Scaler
StringConcatenator	**StringConcatenator**
StringSplitter	**StringSplitter**
DateTransformer	**Date**

*** New ONNX operators are written in red**

- FunctionTransfomer
  - Analyze a lamda function
- FreqnecyEncoder, Aggregator
  - Embed values calculated at training time in a LabelEncoder operator



col_1
A
A
B
A
B

```
fe = FrequencyEncoder(inputs=['col_1',
                              outputs=['col_1_fe'])
fe.fit(training_df)
```

Key	Value
A	3
B	2

"A" appears three times in col_1

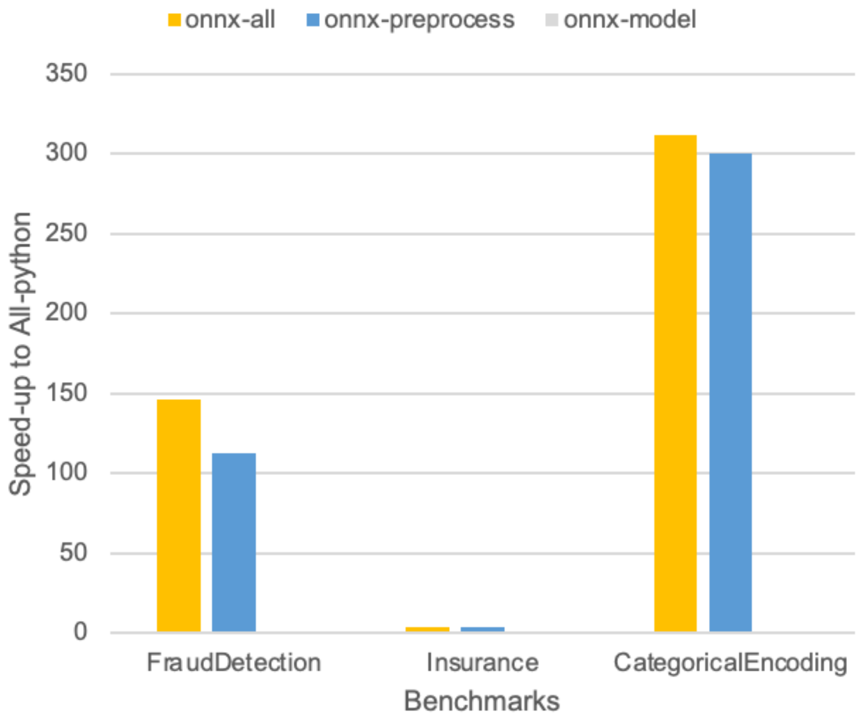
"B" appears two times in col_1

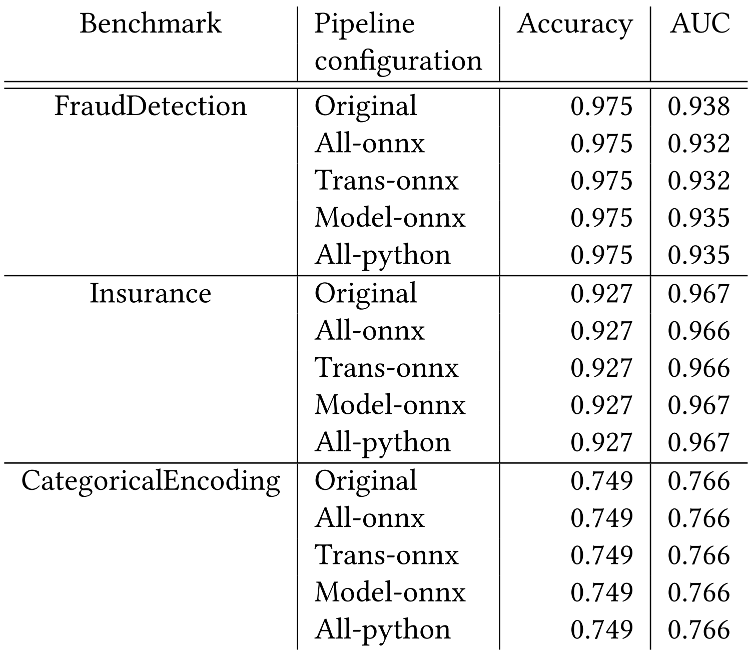
Convert into an ONNX operator

LabelEncoder
keys_strings=['A', 'B']
values_int64s=[3, 2]

Online Scoring Performance

- Up to 300x performance improvement compared to Python
  - Running only a ML model (onnx-model) on the ONNX Runtime did not show much improvement.
- No much difference in prediction accuracy





Benchmark	Pipeline configuration	Accuracy	AUC
FraudDetection	Original	0.975	0.938
	All-onnx	0.975	0.932
	Trans-onnx	0.975	0.932
	Model-onnx	0.975	0.935
	All-python	0.975	0.935
Insurance	Original	0.927	0.967
	All-onnx	0.927	0.966
	Trans-onnx	0.927	0.966
	Model-onnx	0.927	0.967
	All-python	0.927	0.967
CategoricalEncoding	Original	0.749	0.766
	All-onnx	0.749	0.766
	Trans-onnx	0.749	0.766
	Model-onnx	0.749	0.766
	All-python	0.749	0.766