Robust, End-to-end Online Machine Learning Applications with Flyte, Pandera and, Streamlit

PyData Global

Niels Bantilan, Union.ai

10.29.2021

My Background

- 📜 B.A. Biology and Dance
- 📜 M.P.H., Sociomedical Science and Public Health Informatics
- 🦾 Machine Learning Engineer @ Union.ai
- ✈️ Flytekit Maintainer
- ✅ Author and Maintainer of Pandera
- 🛠️ Make DS/ML practitioners more productive

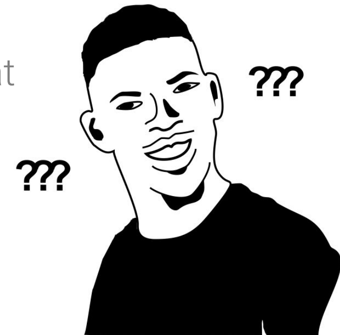
📝 Outline

- ⚡Motivation
- 🤖 Offline vs. Online Learning
- 🌥️ Use Case: Weather Forecasting
- 🏁 Quickstart: Flyte, Pandera, and Streamlit
- 🏗️ Pipeline Architecture
- 💻 Demo
- 🎁 Takeaways

Motivation

"Full-stack Data Scientist / ML engineer"

What even is that



Motivation

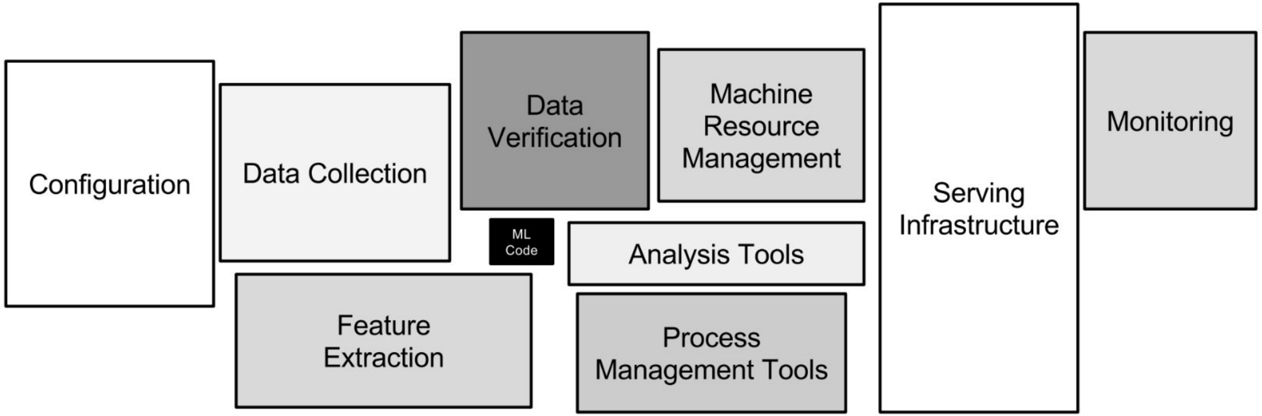
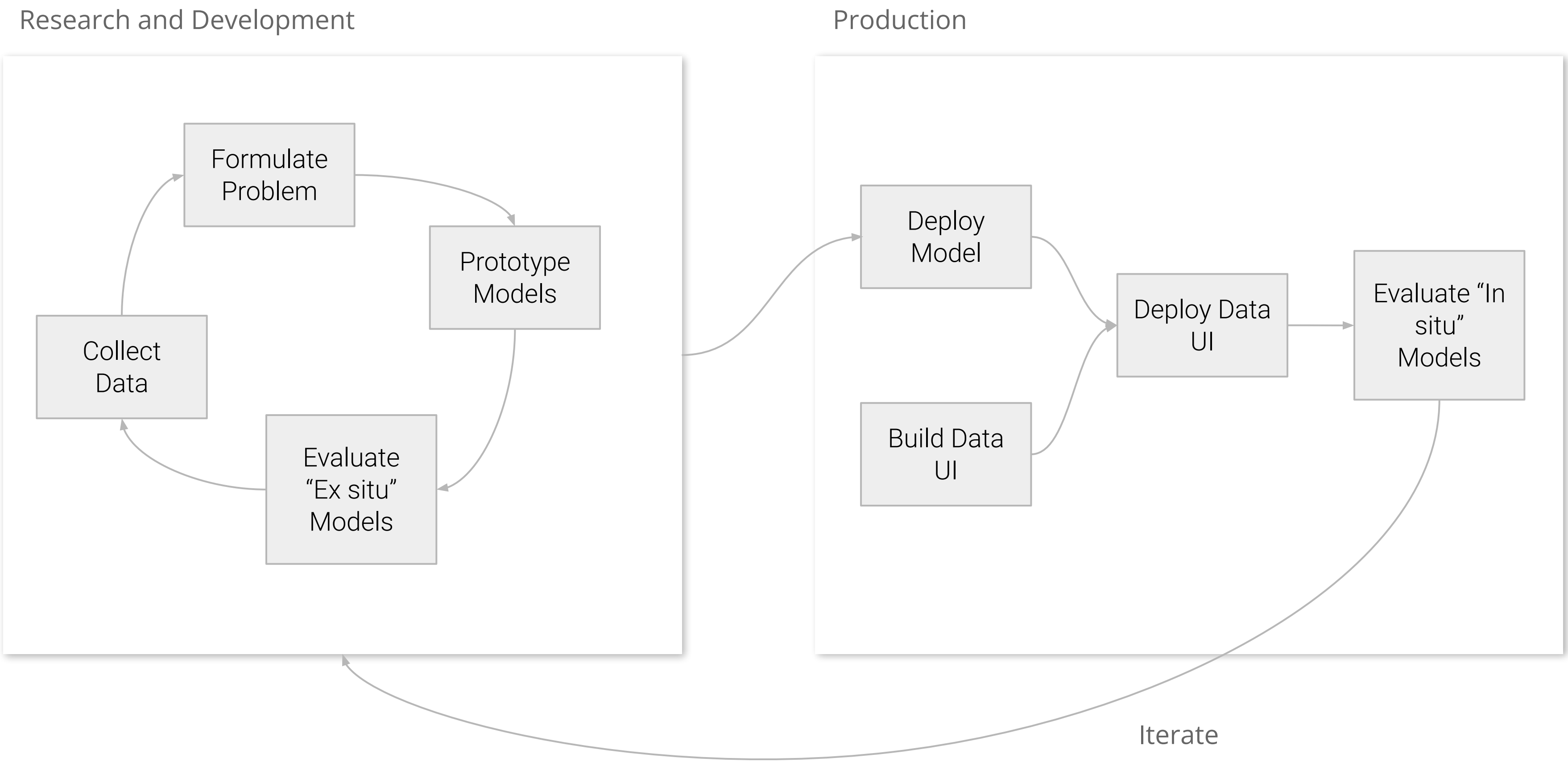


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

source: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

Motivation


Research and Development
Production
Formulate Problem
Prototype Models
Collect Data
Evaluate "Ex situ" Models
Deploy Model
Deploy Data UI
Build Data UI
Evaluate "In situ" Models
Iterate

Motivation

(often-times 1-2 people)

How far can a small data team get building and shipping production model-driven data products?

Challenge

Build an online learning system that updates a model hourly and displays forecasts on a web UI

Why?

It forces the practitioner to think about:

- How to manage incremental data acquisition
- Model updates in production as a first class citizen
- Evaluation metrics where model sees each example only once

Offline vs. Online Learning

A crash course

Flyte

Offline vs. Online Learning

Offline Learning

Learning a model on a static training dataset, potentially with multiple passes through the dataset.

Online Learning

Learning a model with training instances that present themselves to the model only once, in some temporal fashion.

Why Online Learning?

- Your data isn't static, e.g. data is generated as a function of time ⏰
- Your data might be too large to practically store/train a model on all at once
- You can't assume I.I.D. data, e.g. today's 🌡️ depends on yesterday's 🌡️
- Your model needs to be as up-to-date as possible due to the dynamic nature of the data

Use Case: Weather Forecasting

Hourly temperature forecasts

Flyte

Global Hourly - Integrated Surface Database (ISD)

The Integrated Surface Database (ISD) is a global database that consists of hourly and synoptic surface observations compiled from numerous sources into a single common ASCII format and common data model. ISD integrates data from more than 100 original data sources, including numerous data formats that were key-entered from paper forms during the 1950s–1970s time frame. ISD includes numerous parameters such as wind speed and direction, wind gust, temperature, dew point, cloud data, sea level pressure, altimeter setting, station pressure, present weather, visibility, precipitation amounts for various time periods, snow depth, and various other elements as observed by each station.

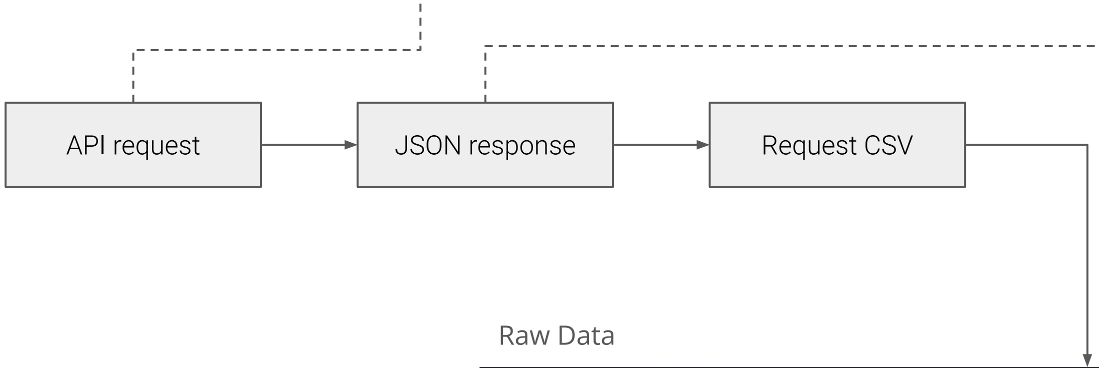
source: https://www.ncei.noaa.gov/products/land-based-station/integrated-surface-database

Use Case: Weather Forecasting

```
https://www.ncei.noaa.gov/access/services/search/v1/data?dataset=global-hourly
&bbox=33.886823,-84.551068,33.647808,-84.28956
&startDate=2021-01-01T00:00:00
&endDate=2021-10-28T16:00:00
&units=metric
&format=json
&limit=1000
&offset=0
```

Index of /data/global-hourly/access/2021

Name	Last modified	Size	Description
Parent Directory		-	
01001099999.csv	2021-10-26 20:00	2.3M	
01001499999.csv	2021-10-26 20:00	1.5M	
01002099999.csv	2021-10-26 20:00	92K	
01003099999.csv	2021-10-26 20:00	211K	
01006099999.csv	2021-10-26 20:00	87K	
01007099999.csv	2021-10-26 20:00	479K	
01008099999.csv	2021-10-26 20:00	5.8M	
01009099999.csv	2021-10-26 20:00	121K	
01010099999.csv	2021-10-26 20:00	6.0M	
01011099999.csv	2021-10-26 20:00	83K	
01015099999.csv	2021-10-26 20:00	1.1M	
01016099999.csv	2021-10-26 20:00	66K	
01017099999.csv	2021-10-26 20:00	70K	
01020099999.csv	2021-10-26 20:00	105K	
01023099999.csv	2021-10-26 20:00	7.1M	
01023199999.csv	2021-10-26 20:00	404	
01024099999.csv	2021-10-26 20:00	87K	
01025099999.csv	2021-10-26 20:00	6.2M	
01026099999.csv	2021-10-26 20:00	2.3M	
01027099999.csv	2021-10-26 20:00	1.6M	



Raw Data

	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	TMP
0	72219013874	2021-01-01T00:00:00	4	33.6301	-84.4418	+0122,1
1	72219013874	2021-01-01T00:52:00	7	33.6301	-84.4418	+0122,5
2	72219013874	2021-01-01T01:52:00	7	33.6301	-84.4418	+0122,5
3	72219013874	2021-01-01T02:07:00	7	33.6301	-84.4418	+0122,5
4	72219013874	2021-01-01T02:19:00	7	33.6301	-84.4418	+0122,5

source: https://www.ncei.noaa.gov/access/search/data-search/global-hourly

Autoregressive Model

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

Actual Implementation: Include one-hot encoded time-based features, e.g. day of week, day of month, month of year, etc.

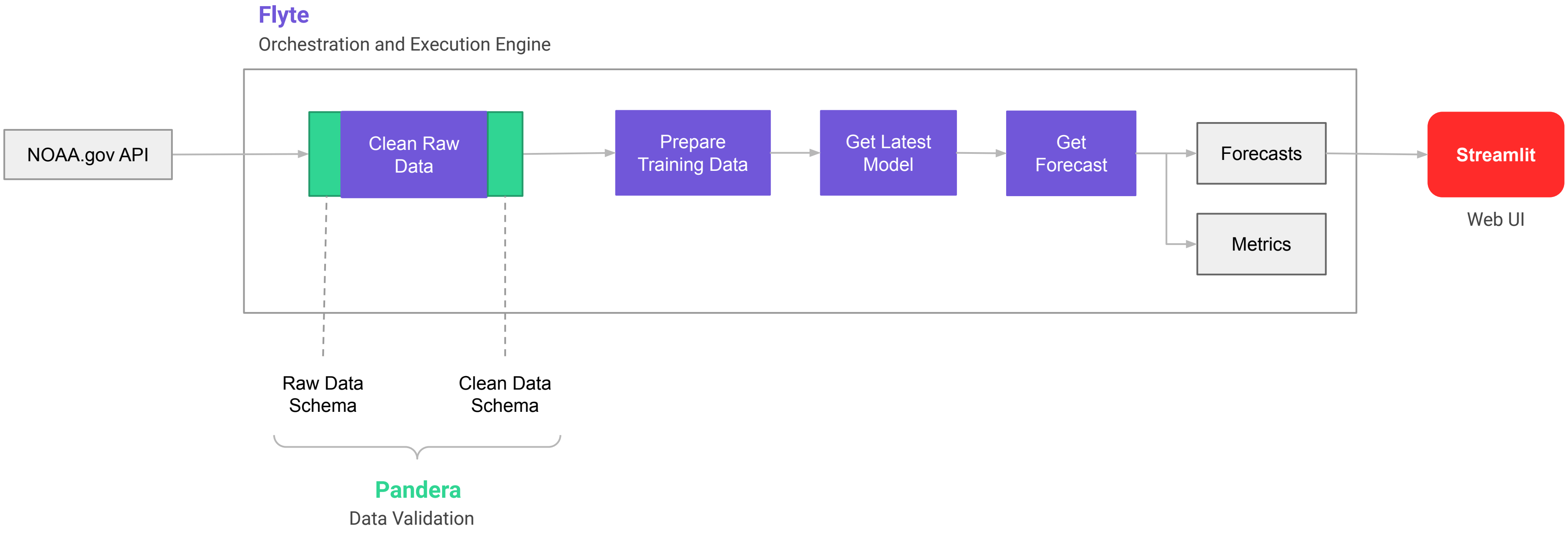
source: https://en.wikipedia.org/wiki/Autoregressive_model

Evaluation Metric

Exponentially-weighted Mean of the Absolute Error

```
eps = 0.1
ewmae = 0

for X, y in training_data:
    y_pred = model(X)
    ewmae = eps * abs(y - y_pred) + (1 - eps) * ewmae
```

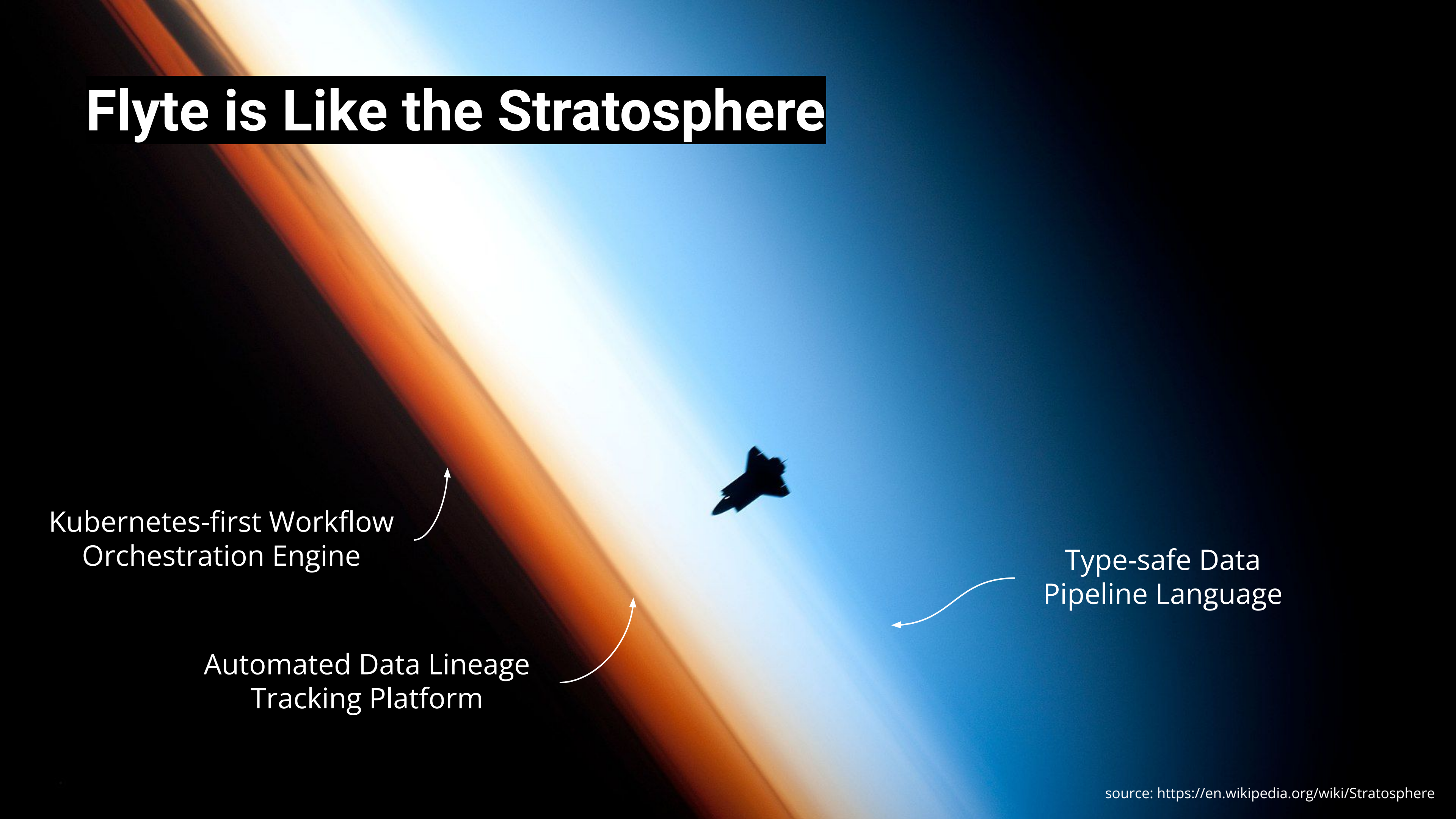
High-level Overview



Quickstart: Flyte

A brief introduction

Flyte is Like the Stratosphere



source: https://en.wikipedia.org/wiki/Stratosphere

Write Tasks: Isolated Pieces of Data Processing

```
from flytekit import task

@task
def transform_data(df: pd.DataFrame) -> pd.DataFrame:
    df["c"] = df["a"] + df["b"]
    df["d"] = df["c"] * 2
    return df

@task
def aggregate_data(df: pd.DataFrame) -> pd.DataFrame:
    return df.mean().to_frame().transpose()
```

Compose Workflows: Connect Tasks Together

```python
from flytekit import workflow

@workflow
def pipeline(df: pd.DataFrame) -> pd.DataFrame:
    transformed_df = transform_data(df=df)
    return aggregate_data(df=transformed_df)
```

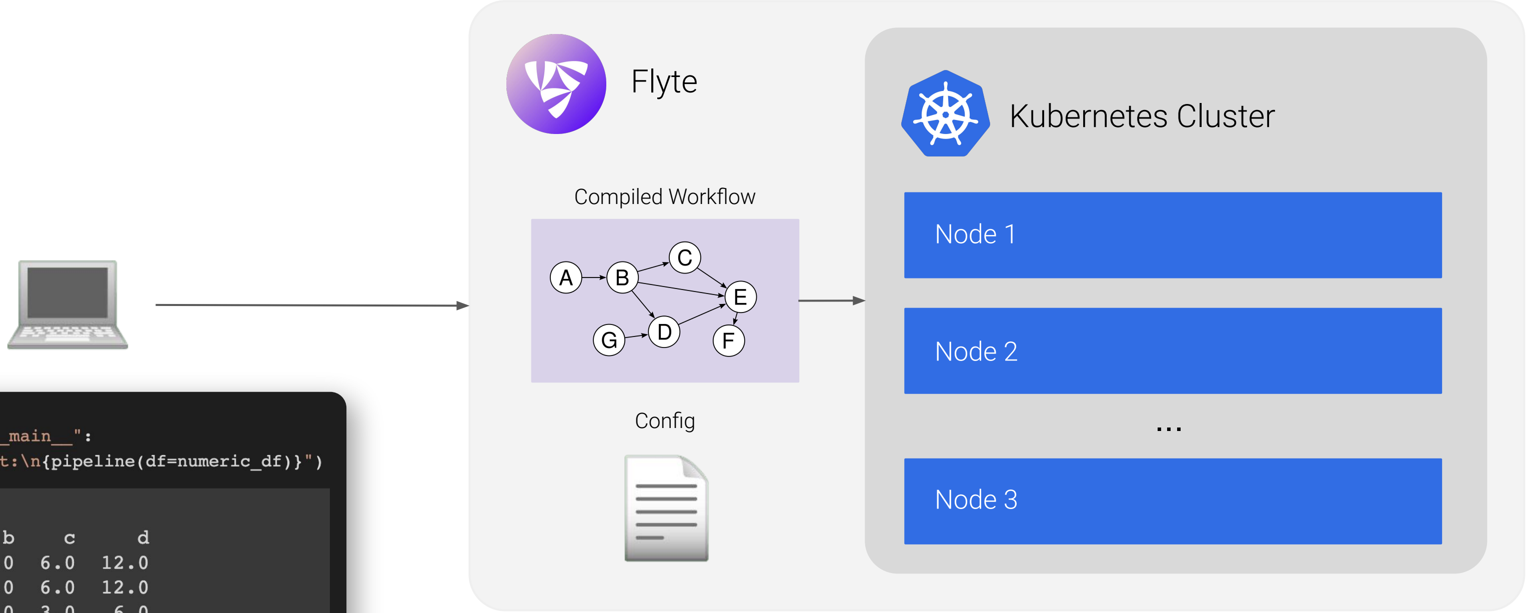
Caching Made Easy

```python
from datetime import datetime

@task(cache=True, cache_version="1.0")
def get_data(from: datetime, to: datetime) -> pd.DataFrame:
    df = pd.read_sql(...). # some expensive query
    return df
```

Ask for Resources, and you Shall Receive

```python
from flytekit import Resources
from flytekit.types.file import FlyteFile
from flytekit.types.directory import FlyteDirectory

@task(
    request_resources=Resources(gpu="2", mem="500Mi", storage="1Gi")
    limit_resources=Resources(gpu="2", mem="750Mi", storage="2Gi")
)
def train_model(training_data: FlyteDirectory) -> FlyteFile:
    ...
```

Develop Locally, Deploy to Scale




```
if __name__ == "__main__":
    print(f"Result:\n{pipeline(df=numeric_df)}")

Result:
          a    b    c     d
mean    2.0  4.0  6.0  12.0
median  2.0  4.0  6.0  12.0
std     1.0  2.0  3.0   6.0
```

Integrate with Existing Ecosystem of Tools

Quickstart: Pandera

A brief introduction

Flyte

Pandera adds guardrails to your pipeline

source: https://time.com/4904297/worlds-longest-suspension-footbridge

Introduction

pandas

MODIN ⭐

Polars

DASK ⭐

Koalas ⭐

RAPIDS

APACHE Spark™

vaex

xarray

⭐ *Coming in the 0.8.0 release*

Encode assumptions about dataframes as schemas

```
import pandera as pa
from pandera.typing import Series, Index

class TransformInput(pa.SchemaModel):
    a: Series[int]
    b: Series[int]

class TransformOutput(TransformInput):
    c: Series[int]
    d: Series[int]

class AggregateOutput(pa.SchemaModel):
    columns: Series[float] = pa.Field(alias="a|b|c|d", regex=True)
    index: Index[str] = pa.Field(isin=["mean", "median", "std"])
```

schema inheritance

regex column-matching

Index support

Extend pandera's built-in validation checks

Custom checks are just class methods

```
class TransformInput(pa.SchemaModel):
    a: Series[int] = pa.Field(ge=0)
    b: Series[int] = pa.Field(ge=0)

    @pa.check("a")
    def check_even(cls, series: Series) -> Series[bool]:
        """Check for even numbers"""
        return series % 2 == 0
```

built-in checks

Custom checks

Easily integrate with your pipeline via function decorators

Use pandera SchemaModels in python type annotations

Validate dataframe types at runtime

```
from pandera.typing import DataFrame

@pa.check_types
def transform_data(df: DataFrame[TransformInput]) -> DataFrame[TransformOutput]:
    df["c"] = df["a"] + df["b"]
    df["d"] = df["c"] * 2
    return df

@pa.check_types
def aggregate_data(df: DataFrame[TransformOutput]) -> DataFrame[AggregateOutput]:
    return df.agg(["mean", "median", "std"])
```

Easily integrates with Flytekit

```
pip install flytekitplugins-pandera
```

```
@task
def transform_data(df: DataFrame[TransformInput]) -> DataFrame[TransformOutput]:
    df["c"] = df["a"] + df["b"]
    df["d"] = df["c"] * 2
    return df
```

Get informative errors if something goes wrong

Column not present in dataframe

```
try:
    transform_data(pd.DataFrame({"b": [4, 5, 6]}))
except pa.errors.SchemaError as e:
    print(e)
```

```
error in check_types decorator of function 'transform_data': column 'a' not in dataframe
   b
0  4
1  5
2  6
```

Get informative errors if something goes wrong

Column has unexpected data type

```
@pa.check_types
def transform_data(df: DataFrame[TransformInput]) -> DataFrame[TransformOutput]:
    df["c"] = df["a"] + df["b"]
    df["d"] = df["c"] * 2
    return df

try:
    transform_data(pd.DataFrame({"a": ["x", "y", "z"], "b": [4, 5, 6]}))
except pa.errors.SchemaError as e:
    print(e)
```

```
error in check_types decorator of function 'transform_data': expected series 'a' to have type int64, got object
```

Get informative errors if something goes wrong

Assumptions violated: presence of negative numbers and odd numbers

```
@pa.check_types(lazy=True)
def transform_data(df: DataFrame[TransformInput]) -> DataFrame[TransformOutput]:
    df["c"] = df["a"] + df["b"]
    df["d"] = df["c"] * 2
    return df

try:
    transform_data(pd.DataFrame({"a": [-1,2,3], "b": [4,3,-6]}))
except pa.errors.SchemaErrors as e:
    display(e.failure_cases)
```

Inspect failure cases

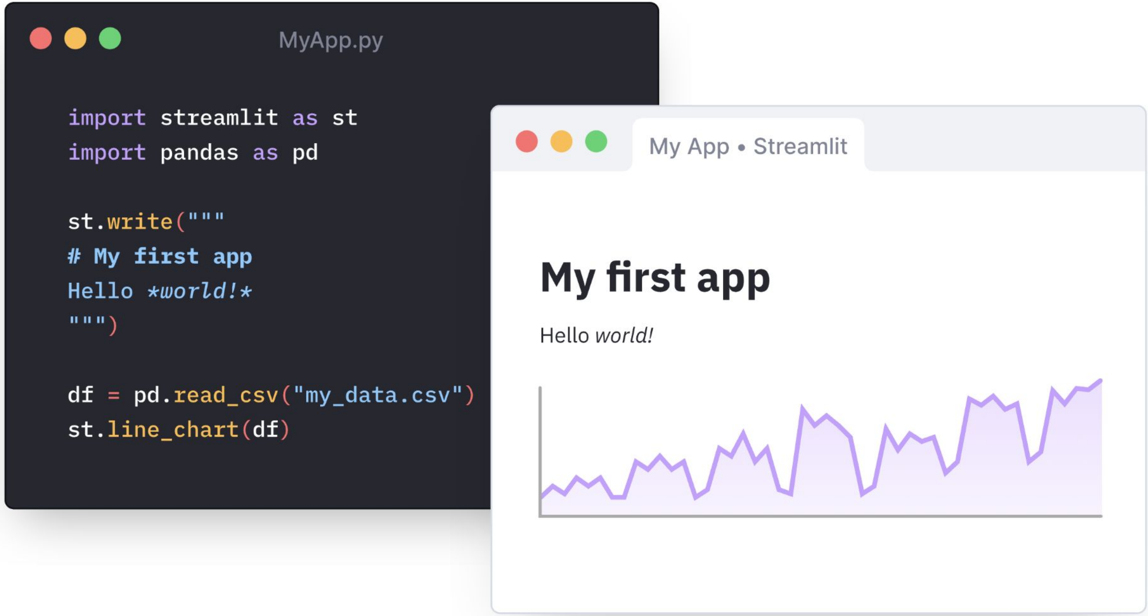
	schema_context	column	check	check_number	failure_case	index
0	Column	a	greater_than_or_equal_to(0)	0	-1	0
1	Column	a	check_even	1	-1	0
2	Column	a	check_even	1	3	2
3	Column	b	greater_than_or_equal_to(0)	0	-6	2

Quickstart: Streamlit

A brief introduction

Flyte

Make Interactive UIs via Scripting



Use Case: Weather Forecasting

Streamlit 🤝 Flyte

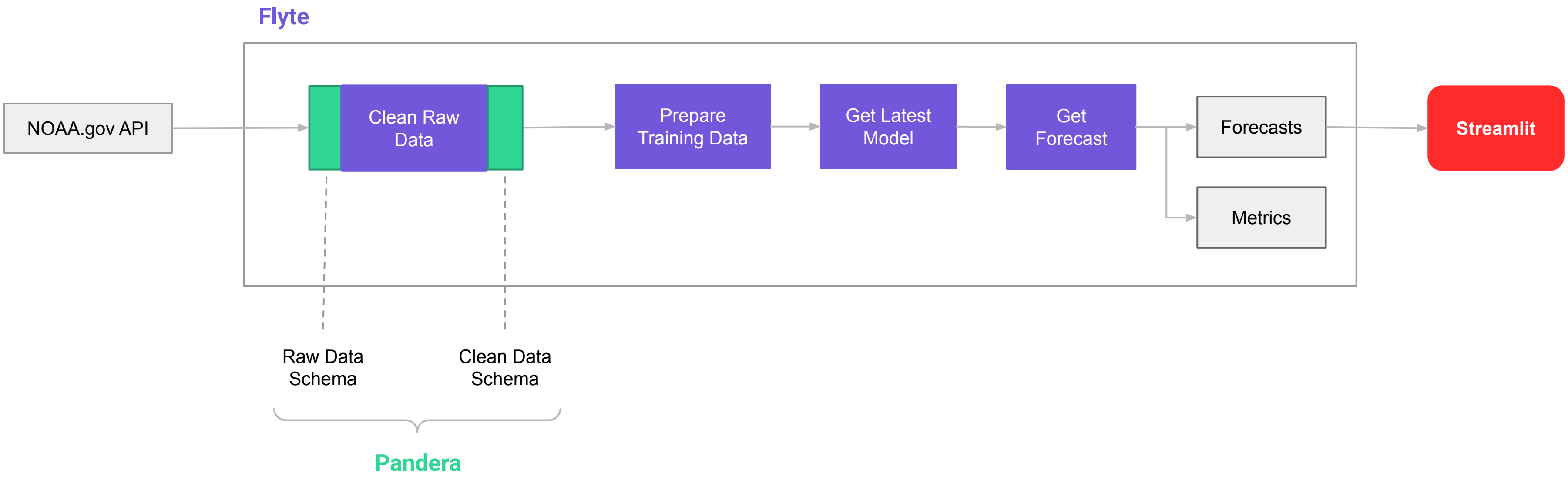
```
import pandas as pd
import streamlit as st

from flytekit.remote import FlyteRemote

remote = FlyteRemote()

st.title("My Flyte App")

def get_flyte_output():
    [latest_execution, *_], _ = remote.client.list_executions_paginated(...)
    execution_output: pd.DataFrame = remote.client.get_execution_data(latest_execution.id)
    return execution_output

st.linechart(get_flyte_output())
```

Pipeline Architecture

Putting all the Pieces Together

Use Case: Weather Forecasting

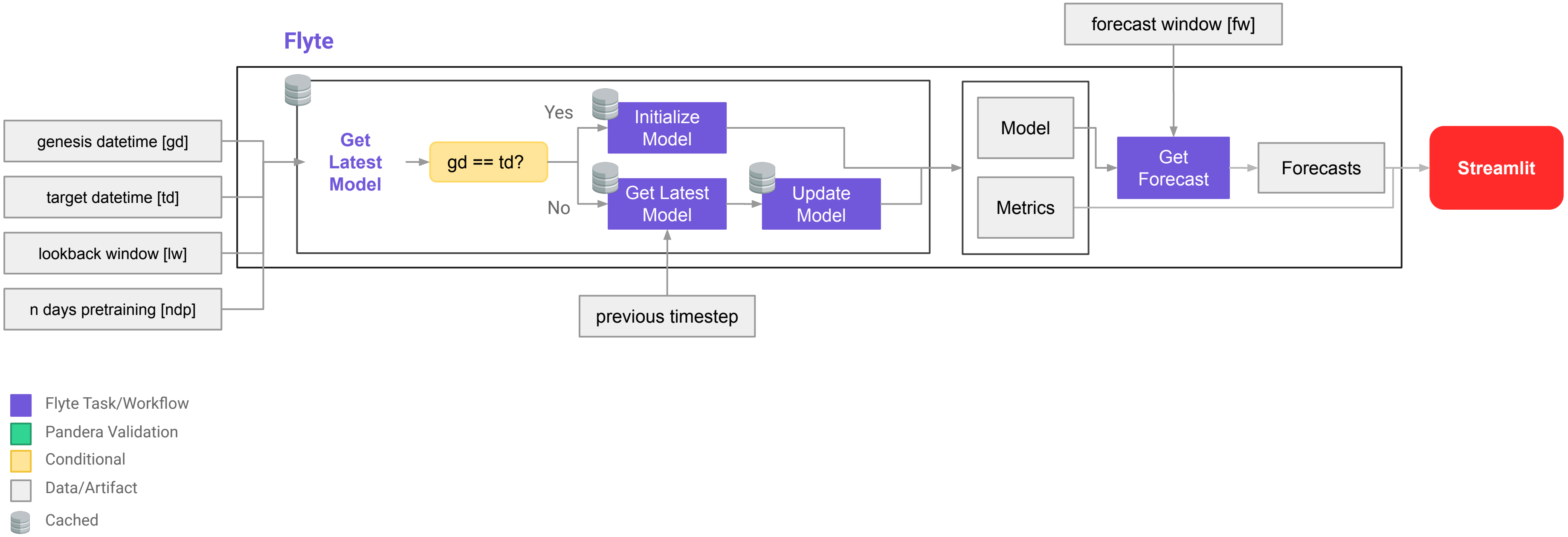
High-level Overview



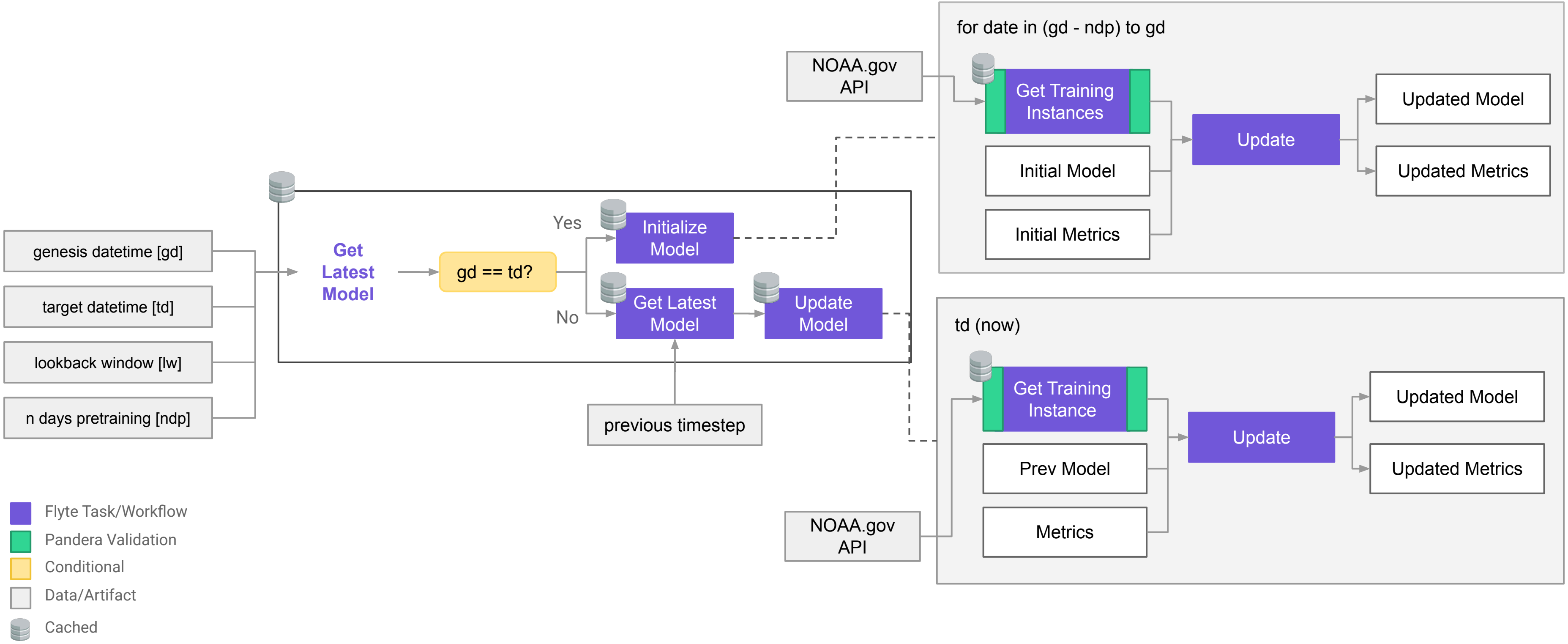
Requirements

- Support incremental model updates and pre-training
- Don’t waste requests fetching data from the NOAA API
- Implement online evaluation metrics

Support Incremental Model Updates and Pretraining



Don't Waste Requests Fetching Data from NOAA API



Online Model Evaluation

Exponentially-weighted Mean of the Absolute Error

```
train_ewmae = 0
valid_ewmae = 0
eps = 0.1

for X, y in training_data:
    # update validation error before model update
    y_pred = model(X)
    valid_ewmae = eps * abs(y - y_pred) + (1 - eps) * valid_ewmae

    # update the model
    model.fit_partial(X, y)

    # update the training error
    y_pred = model(X)
    train_ewmae = eps * abs(y - y_pred) + (1 - eps) * train_ewmae
```

Demo

Web UI and Flyte Console

https://bit.ly/flyte-weather-forecasting

Takeaways

Towards “Full-stack Data Science and ML engineering” teams

Flyte

Flyte provides a powerful platform for type-safe, scalable, and reproducible model-driven applications

Pandera

Pandera enables type safety for dataframe-like objects through an expressive, lightweight API

Streamlit

Streamlit simplifies and accelerates web UI development for anyone who can script

Questions

Contact

twitter: @cosmicBboy
github: cosmicBboy
linkedin: https://www.linkedin.com/in/nbantilan

Links

Weather Forecasting app: https://bit.ly/flyte-weather-forecasting
Weather Forecasting repo: https://github.com/flyteorg/flytelab
Flyte repo: https://github.com/flyteorg/flyte
Pandera repo: https://github.com/pandera-dev/pandera
Streamlit repo: https://github.com/streamlit/streamlit

Flyte