Taming the black box

How to monitor Machine Learning models in production

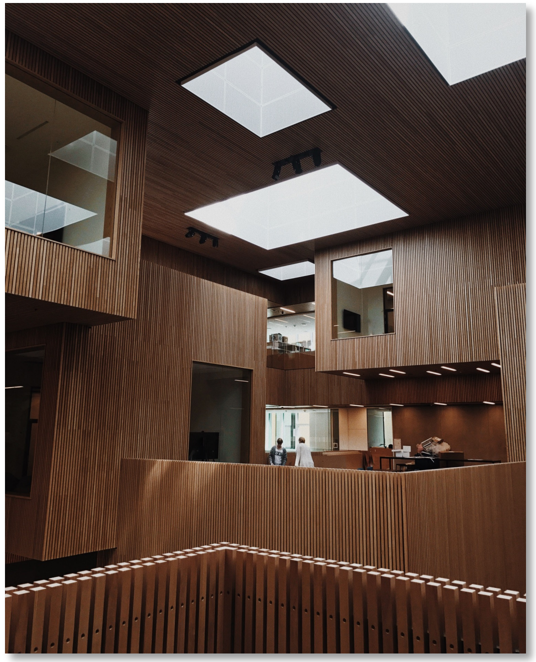
MLOps Jan 2023 - DTU

Duarte O.Carmo

duarteocarmo.com - @duarteocarmo

Who even are you?

- */du-art/*
- DTU graduated in 2018 (Eng. Management LOL)
- ML/Software Engineer - Contractor
- From Lisbon, based in Copenhagen
- *Past:* Strategy, Product Management, New Ventures, Management Consulting
- I write code and solve problems end-to-end
- I like running a lot



This is a lecture to show you how to deal with things ~~if~~ <u>when</u> they break

- Talk about failures
- Things to keep in mind
- How to see/prevent failures

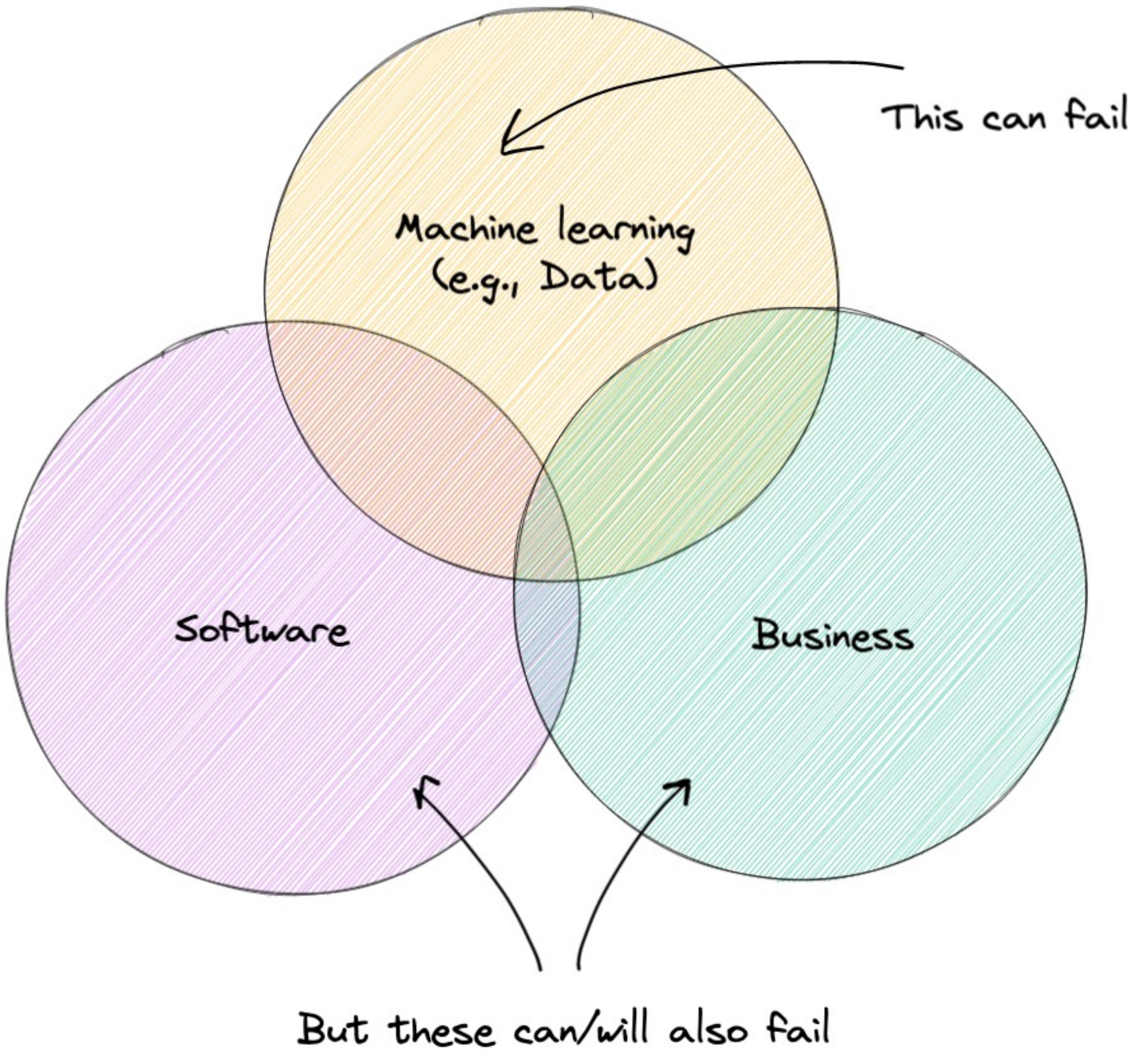
- This is a new field
- None of this is set in stone
- Life is made of tradeoffs

DTU

1 | What can fail?

ML applications will fail in a **myriad** of ways

(but we can group them in 3)



DTU

1.1 | Software failures

1.2 | ML failures

1.3 | Business failures

Software is never done

(only abandoned)

All of the reasons your non-ML application can already fail

- Dependencies
- Deployments
- Hardware
- Downtime/crashing

1.1 | Software failures

1.2 | ML failures

1.3 | Business failures

Failure 1 | Can you handle an edge case?

- Text classifier receives an empty string
- You don’t receive an int
- Self driving car gets stopped by Police
- Driving in US != Driving in Malaysia

DTU

Failure 2 | Degenerate feedback loops – when predictions influence feedback

- Recommendation systems
- YouTube/Spotify/Netflix algorithm
- Filter bubble
- TikTok and randomization

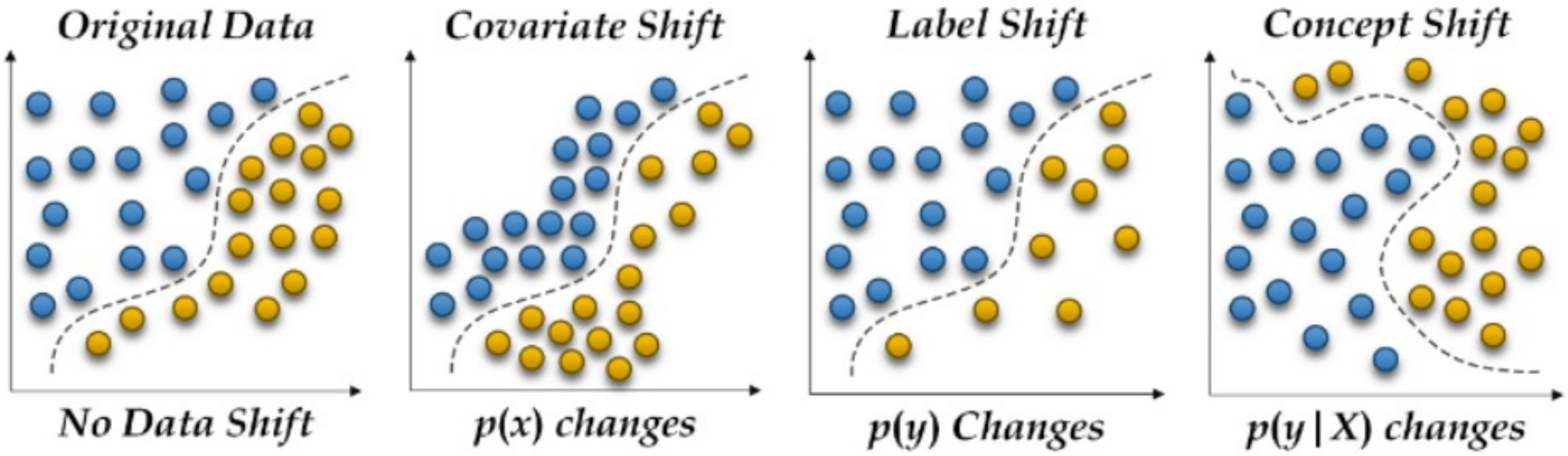
"My desire to be well-informed is currently at odds with my desire to remain sane."

Failure 3 |

training != production

x : your inputs

y: your outputs



(a) Original data (b) Covariate shift (c) Label shift (d) Concept shift

"World" | Your inputs change | Your outputs change | Their relation changes

Data Drift

The model performs worse on unknown data regions

Target Drift

The world has changed, and you need to wake up

Data Drift

The model performs worse on unknown data regions

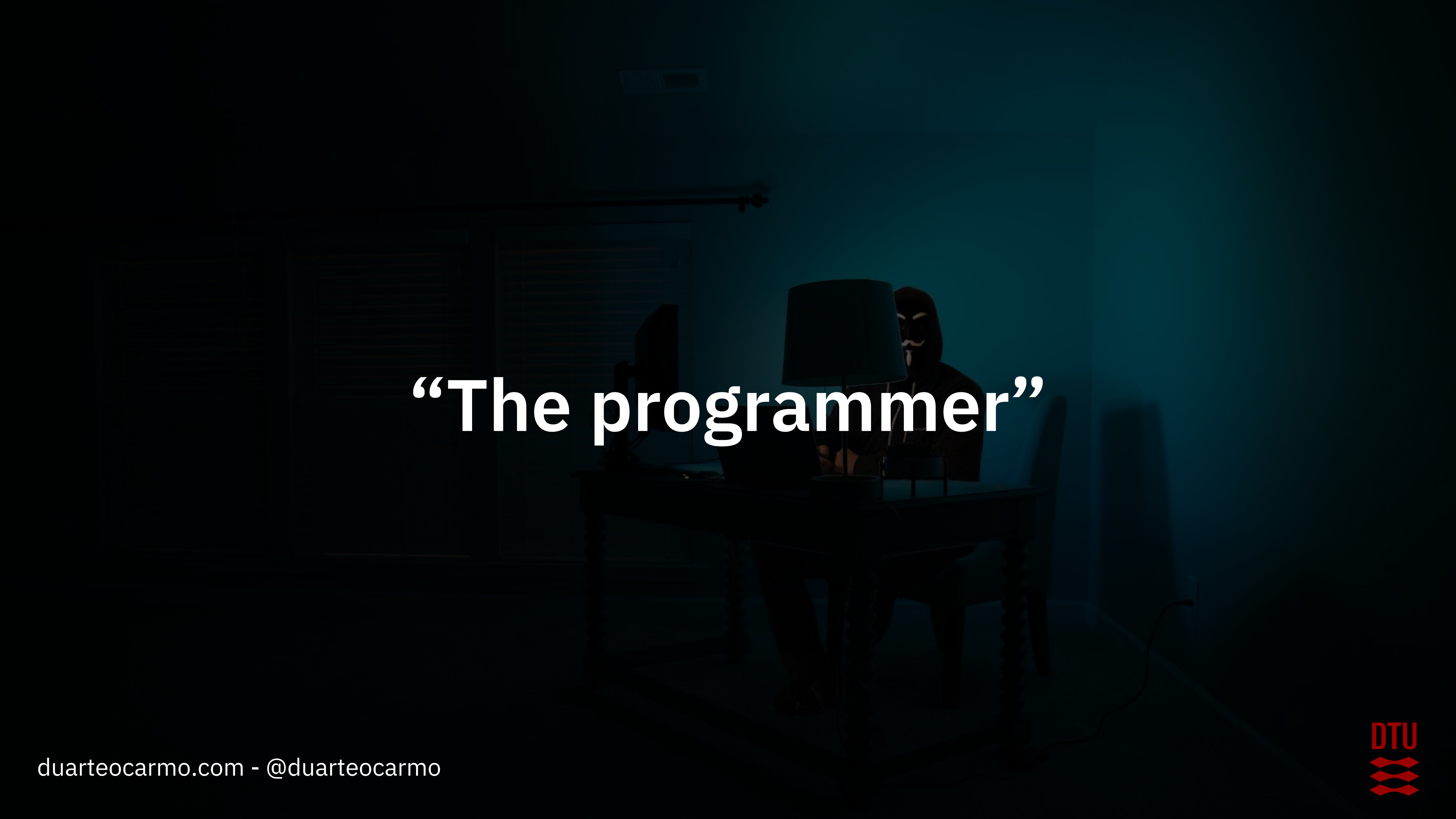
Target Drift

The world has changed, and you need to wake up

How to address

1. Train the model on a massive dataset
2. Domain adaptation (experimental – google it)
3. Retrain your model from scratch, or from the last checkpoint

1.1 | Software failures
1.2 | ML failures
1.3 | Business failures



“The programmer”

How **many** predictions are we making?

How is the **KPI** evolving?

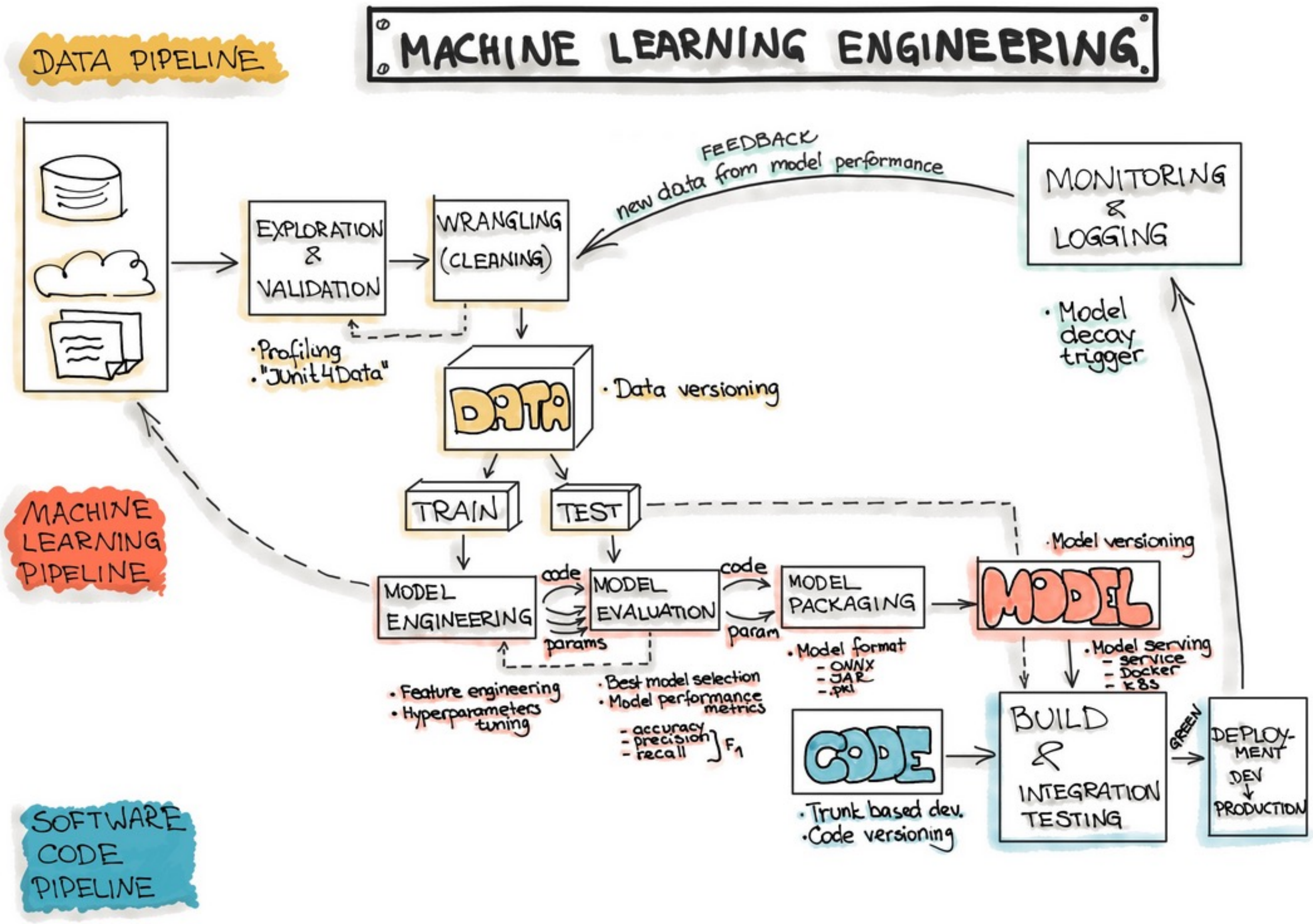
Do you mind **also** predicting X?

Ah really? **I thought** Y was happening..

How much **value** are you delivering?

Why did you predict Y on X?

2 | Before you make a model



Credits: ml-ops.org

There are 3 mains ways of knowing how your model is performing in production

Hand labels

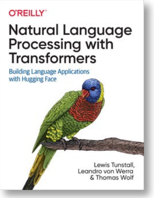
- Annotate labels by hand
- It can get expensive
- Models require less (e.g., fine tunning)

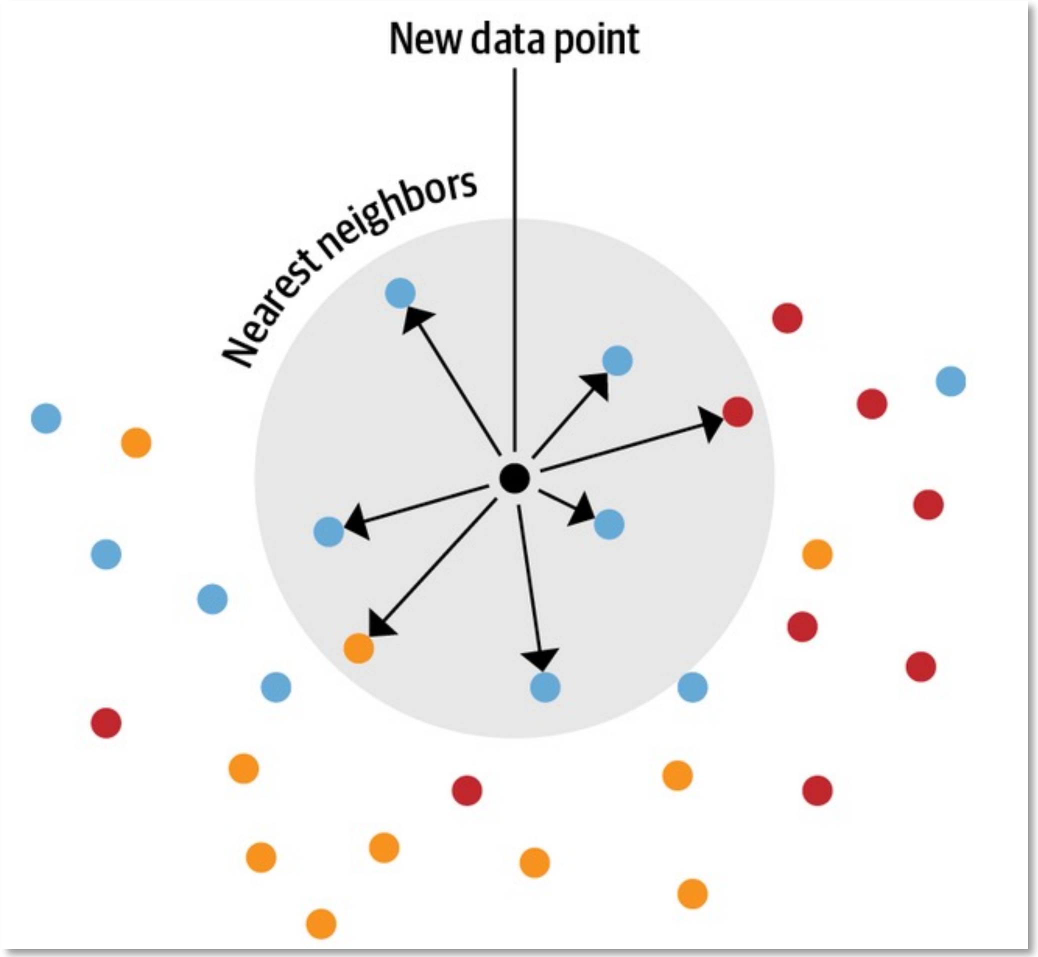
Natural labels

- You know your performance in production
- Trip prediction, forecasting, timeseries
- Ensure system to leverage them

Programmatic labels

- No natural labels
- Recommendation not clicked is a bad label
- Get creative (thumbs up, copy paste, user feedback)





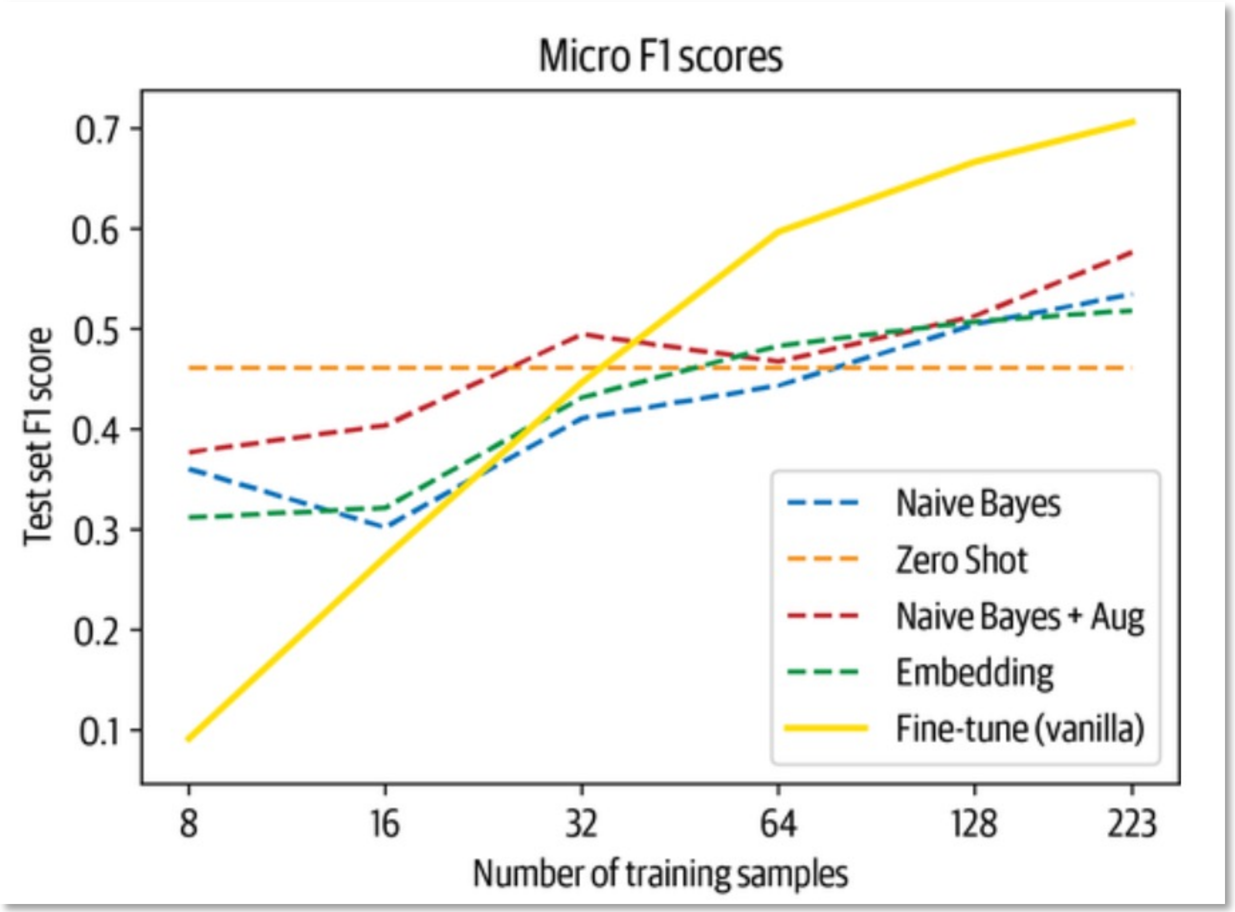
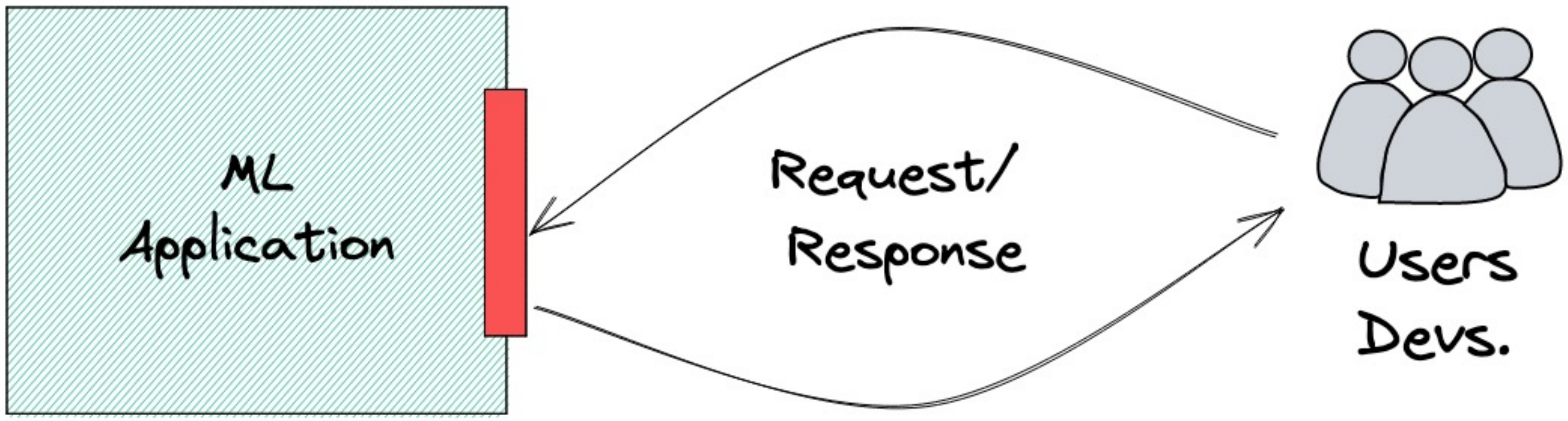


Figure 1: Making a lot with a Little
Credits: Lewis Tunstall, NLP with Transformers O'Reilly

Figure 2: Nearest neighbour lookup
Credits: Lewis Tunstall, NLP with Transformers O'Reilly

3 | Let’s get practical


ML
Application
Request/
Response
Users
Devs.

3.1 | Software Monitoring
3.2 | ML Monitoring
3.3 | Business reporting

What the hell is OpenTelemetry?

```python
from opentelemetry import trace


current_span = trace.get_current_span()


current_span.set_attribute("operation.value", 1)
current_span.set_attribute("operation.name", "Saying hello!")
current_span.set_attribute("operation.other-stuff", [1, 2, 3])
```

```python
import fastapi
from opentelemetry import trace
from opentelemetry.exporter.otlp.proto.http.trace_exporter import (
    OTLPSpanExporter,
)
from opentelemetry.instrumentation.fastapi import FastAPIInstrumentor
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import BatchSpanProcessor


from .models import Result, Item


provider = TracerProvider()
processor = BatchSpanProcessor(OTLPSpanExporter())
provider.add_span_processor(processor)
trace.set_tracer_provider(provider)
tracer = trace.get_tracer(__name__)


app = fastapi.FastAPI(title="demo")
FastAPIInstrumentor.instrument_app(app)


@app.post("/predict/", reponse_model=Result)
def predict(features: Item):
    current_span = trace.get_current_span()
    input_hash = hash(features)
    current_span.set_attribute("app.demo.input_hash", input_hash)
    prediction = get_prediction_for(features)
    current_span.set_attribute("app.demo.prediction", prediction)
    return prediction


@app.post("/feedback")
def receive_feedback(request):
    current_span = trace.get_current_span()
    save_to_db(request.feedback)
    current_span.set_attribute("app.demo.feedback", request.feedback)
    return {"received": "ok"}
```

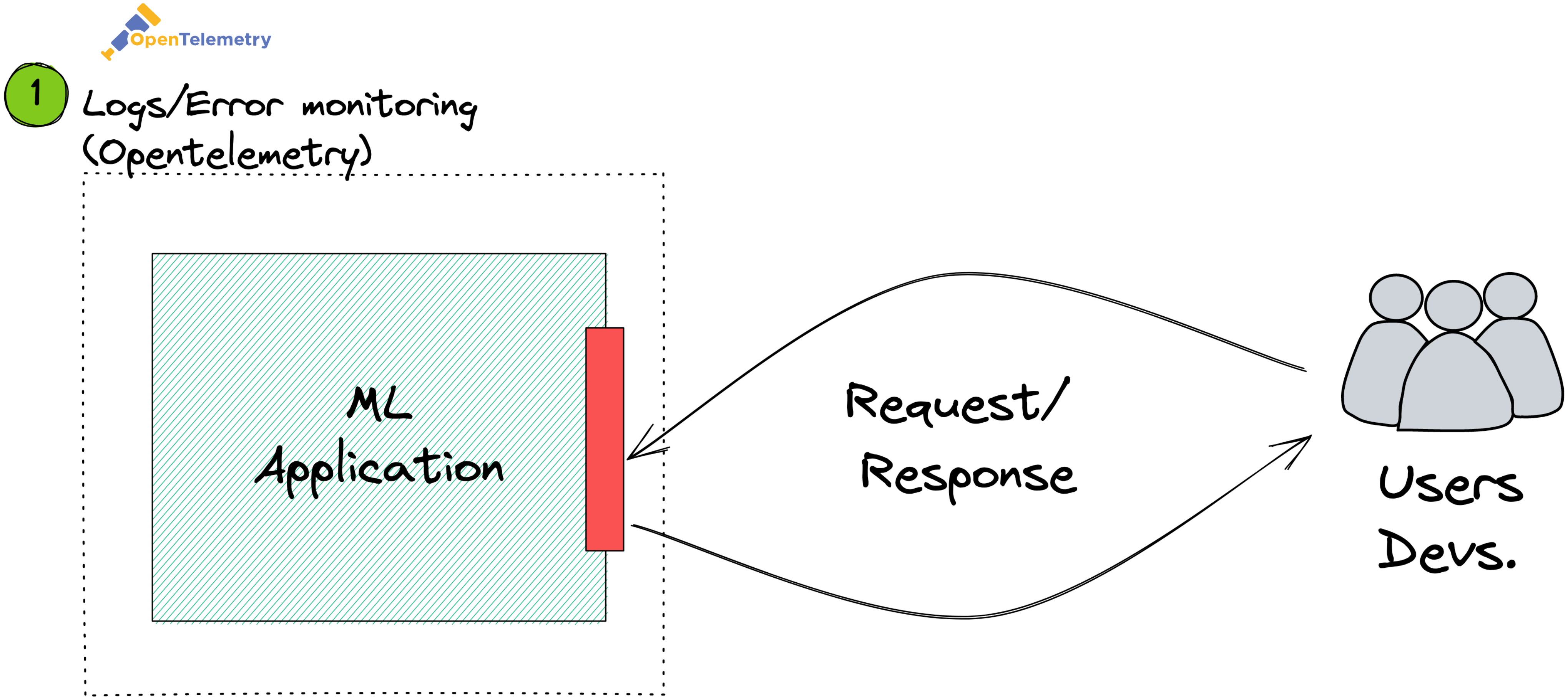
Import opentelemetry and FastAPIInstrumentor

Initialize instrumentation

Initialize FastAPI

Save prediction to opentelemetry

Don't forget feedback


OpenTelemetry
1
Logs/Error monitoring
(Opentelemetry)
ML
Application
Request/
Response
Users
Devs.

3.1 | Software Monitoring
3.2 | ML Monitoring
3.3 | Business reporting

1.Save your predictions to a database

```python
# monitoring.py
# ...
def save_to_database(input: Item, result: Result) -> None:
    """
    Saves input/output dicts to bigquery
    """
    client = BigQuery.client()
    table = "your_cool_bq_table"
    current_time = datetime.datetime.now()

    rows_to_insert = [(current_time, input.json(), result.json())]
    errors = client.insert_rows(table,
                                rows_to_insert)

    if errors:
        logging.info(f"Error: {str(errors)}")
        return

    logging.info("Saved prediction")
```

2.Don't block responses with saving

```python
# app.py
# ...
from fastapi import FastAPI, BackgroundTasks
from .monitoring import save_to_database
# ...

# create an endpoint that receives POST requests
@app.post("/predict/",
          reponse_model=Result,
          background_tasks: BackgroundTasks)
def predict(features: Item):
    # some processing
    prediction = get_prediction_for(features)
    background_tasks.add_task(save_to_bq, input=features, result=prediction)
    return prediction
```

3.Load reference and predicted data

```python
# ... rest of the monitoring.py

DATA_WINDOW_SIZE = 3000 # how many predictions to load

# loads our training/reference dataset
def load_train_data() -> pandas.DataFrame:
    train_file = "static/train_data.csv"
    train_df = pandas.read_csv(train_file)
    return train_df

# loads our latest predictions
def load_last_predictions() -> pandas.DataFrame:
    query = f"""
    SELECT created_at, input, output
    FROM `my_cool_bgq_table`
    ORDER BY created_at DESC
    LIMIT {DATA_WINDOW_SIZE};
    """
    prediction_data = pandas.read_gbq(query=query)
    return prediction_data
```

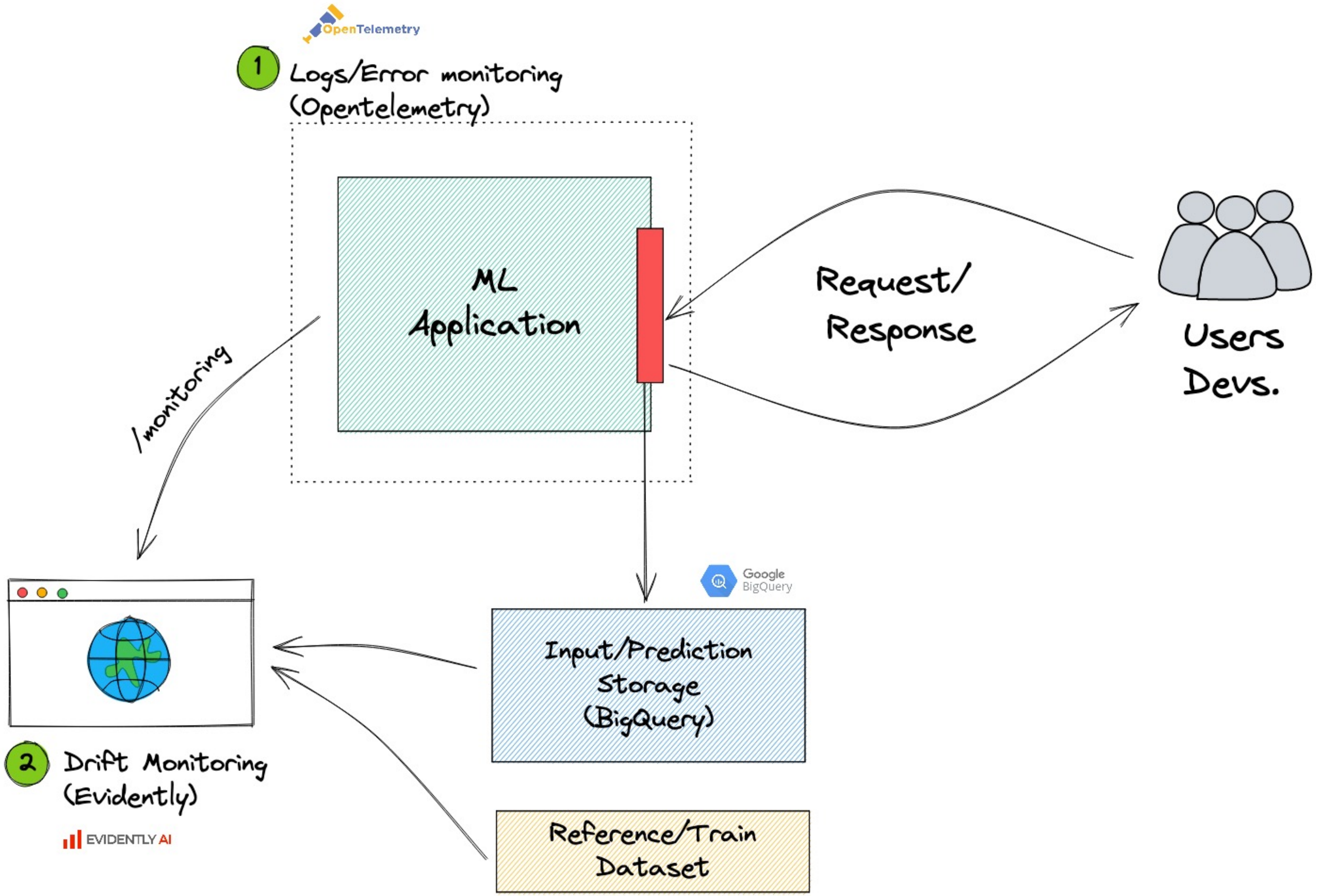

DTU

4.Generate your dashboard

```python
# ... rest of the monitoring.py

# this function generates a dashboard from our reference and prediction data
# which is then saved to a `drift.html` file
def generate_dashboard() -> str:
    dasboard_name = "static/drift.html"
    data_drift_dashboard = Dashboard(
        tabs=[
            DataDriftTab(verbose_level=0),
        ]
    )

    reference_data = load_reference_data()
    current_data = load_last_predictions()

    data_drift_dashboard.calculate(
        reference_data=reference_data,
        current_data=current_data,
        column_mapping=None,
    )

    data_drift_dashboard.save(dasboard_name)
    logger.info(f"Dashboard saved to {dasboard_name}")
    return dasboard_name
```

5.Serve your dashboard

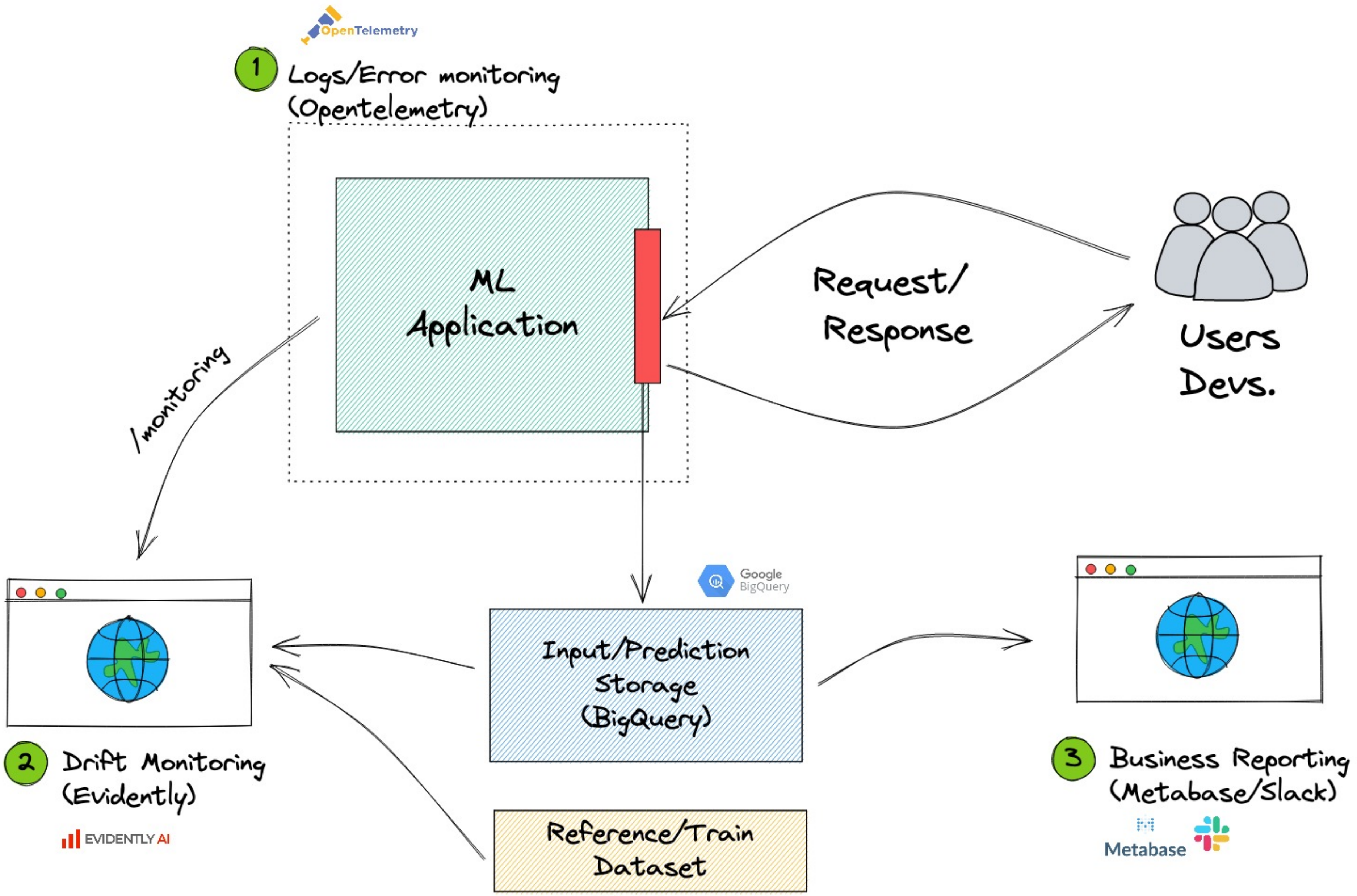
```python
from .monitoring import generate_dashboard
# ... rest of the main.py

@app.get("/monitoring", tags=["Other"])
def monitoring():
    dashboard_location = generate_dashboard()
    return FileResponse(dashboard_location)
```


OpenTelemetry
1
Logs/Error monitoring
(Opentelemetry)
ML
Application
Request/
Response
Users
Devs.
/monitoring
Google
BigQuery
Input/Prediction
Storage
(BigQuery)
2
Drift Monitoring
(Evidently)
EVIDENTLY AI
Reference/Train
Dataset

DTU

Every company has a BI tool

Metabase
Supercell
PowerBI
Excel
Sheets
Slack
...



OpenTelemetry
1
Logs/Error monitoring
(Opentelemetry)
ML
Application
Request/
Response
Users
Devs.
/monitoring
Google
BigQuery
Input/Prediction
Storage
(BigQuery)
Reference/Train
Dataset
2
Drift Monitoring
(Evidently)
EVIDENTLY AI
3
Business Reporting
(Metabase/Slack)
Metabase

What's the point in the end?

1. Things break – know how
2. You're not working in a basement
3. Get feedback, and save your predictions
4. Don't focus on the tool, focus on the task

Thank you, questions?

Feedback: tinyurl.com/duarte-lecture