WELCOME

AIRSPACE
INTELLIGENCE

AIRSPACE INTELLIGENCE

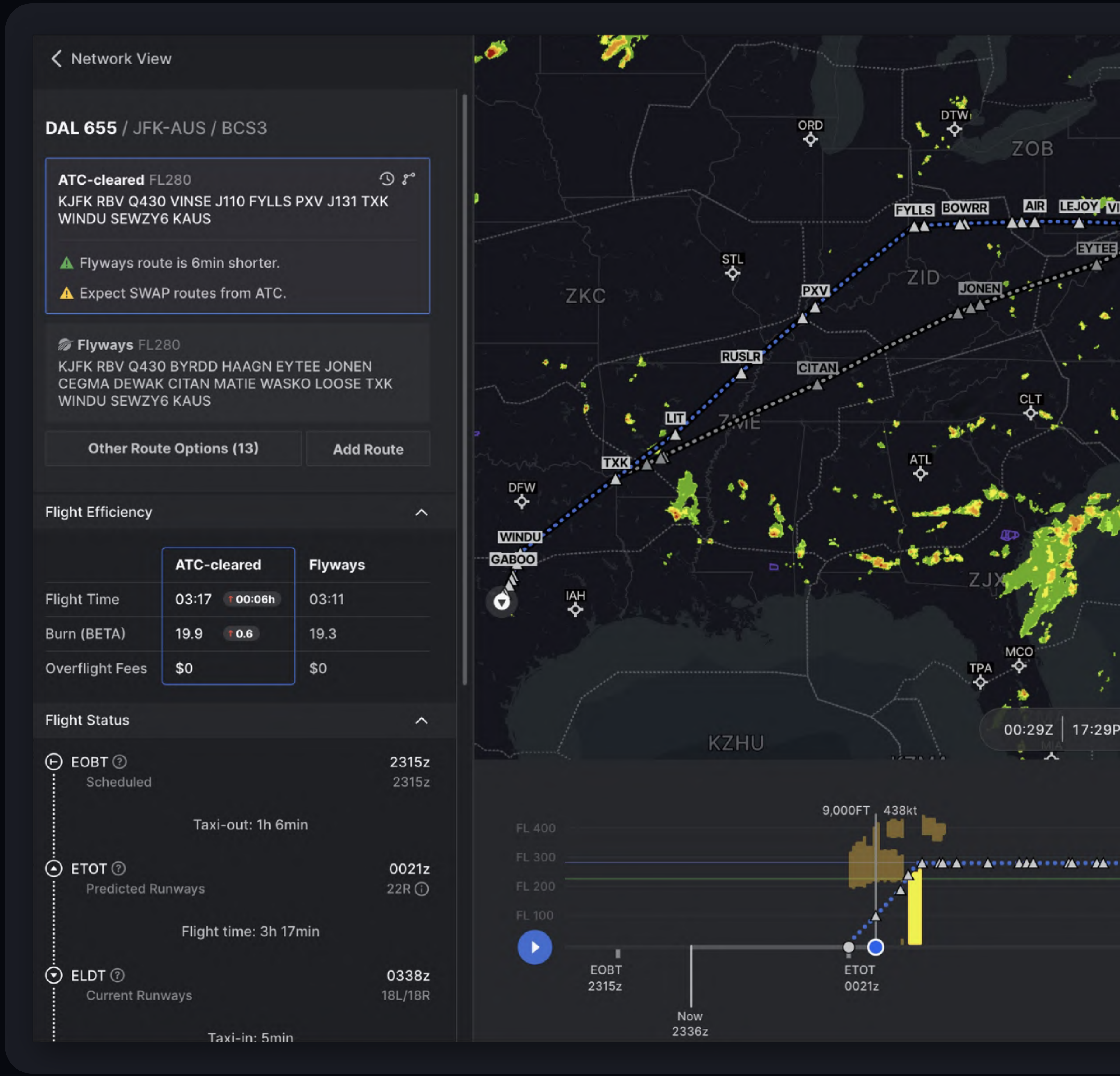
PRODUCTION ML - BEST PRACTICES

PIOTR MAZUR, SENIOR ML ENGINEER & TEAM LEAD

FLYWAYS AI™ PLATFORM

The operating system for airline network operations.

Flyways is an AI-enabled platform designed to accelerate & improve operational decision making capabilities in the world's most complex and dynamic airspace.



DAL 655 / JFK-AUS / BCS3

ATC-cleared FL280
KJFK RBV Q430 VINSE J110 FYLLS PXV J131 TXK WINDU SEWZY6 KAUS

⚠ Flyways route is 6min shorter.
⚠ Expect SWAP routes from ATC.

Flyways FL280
KJFK RBV Q430 BYRDD HAAGN EYTEE JONEN CEGMA DEWAK CITAN MATIE WASKO LOOSE TXK WINDU SEWZY6 KAUS

Other Route Options (13) | Add Route

Flight Efficiency

	ATC-cleared	Flyways
Flight Time	03:17 ↑00:06h	03:11
Burn (BETA)	19.9 ↑0.6	19.3
Overflight Fees	$0	$0

Flight Status

EOBT ⓘ 2315z
Scheduled 2315z

Taxi-out: 1h 6min

ETOT ⓘ 0021z
Predicted Runways 22R ⓘ

Flight time: 3h 17min

ELDT ⓘ 0338z
Current Runways 18L/18R

Taxi-in: 5min

Company

Silicon Valley know-how for airspace operations.

We are a team of multi-domain experts from **Google, Amazon, Lockheed Martin, Boeing, Palantir** and **the Federal Aviation Administration**. Airspace Intelligence is well-funded by venture capital from the leading investors and institutions for artificial intelligence.

Investors

RENEGADE

Google

FRANKLIN TEMPLETON INVESTMENTS

Bloomberg BETA

Stanford University

xyzcapital

Greg Brockman
CTO @ Open AI

Di-Ann Eisnor
Waze

Zak Stone
Google Brain

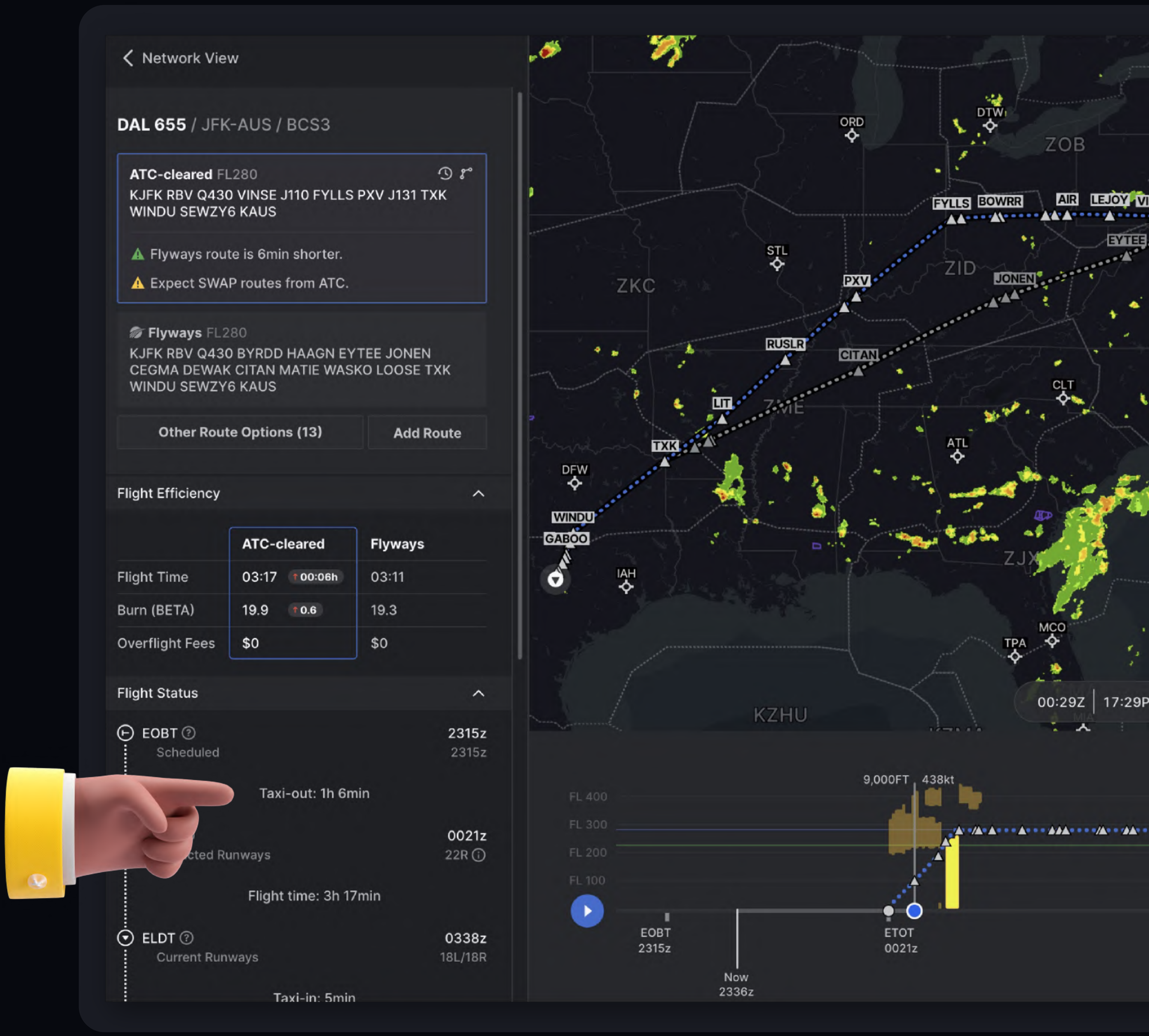
Anthony Goldbloom
CEO @ Kaggle

Piotr Mazur
Senior ML Engineer & Team Lead
Airspace Intelligence

FLYWAYS AI™ PLATFORM

The operating system for airline network operations.

Flyways is an AI-enabled platform designed to accelerate & improve operational decision making capabilities in the world's most complex and dynamic airspace.



Network View

DAL 655 / JFK-AUS / BCS3

ATC-cleared FL280
KJFK RBV Q430 VINSE J110 FYLLS PXV J131 TXK WINDU SEWZY6 KAUS

Flyways route is 6min shorter.

Expect SWAP routes from ATC.

Flyways FL280
KJFK RBV Q430 BYRDD HAAGN EYTEE JONEN CEGMA DEWAK CITAN MATIE WASKO LOOSE TXK WINDU SEWZY6 KAUS

Other Route Options (13) | Add Route

Flight Efficiency

	ATC-cleared		Flyways
Flight Time	03:17	00:06h	03:11
Burn (BETA)	19.9	0.6	19.3
Overflight Fees	$0		$0

Flight Status

EOBT Scheduled 2315z 2315z

Taxi-out: 1h 6min

cted Runways 0021z 22R

Flight time: 3h 17min

ELDT Current Runways 0338z 18L/18R

Taxi-in: 5min

Data

Data

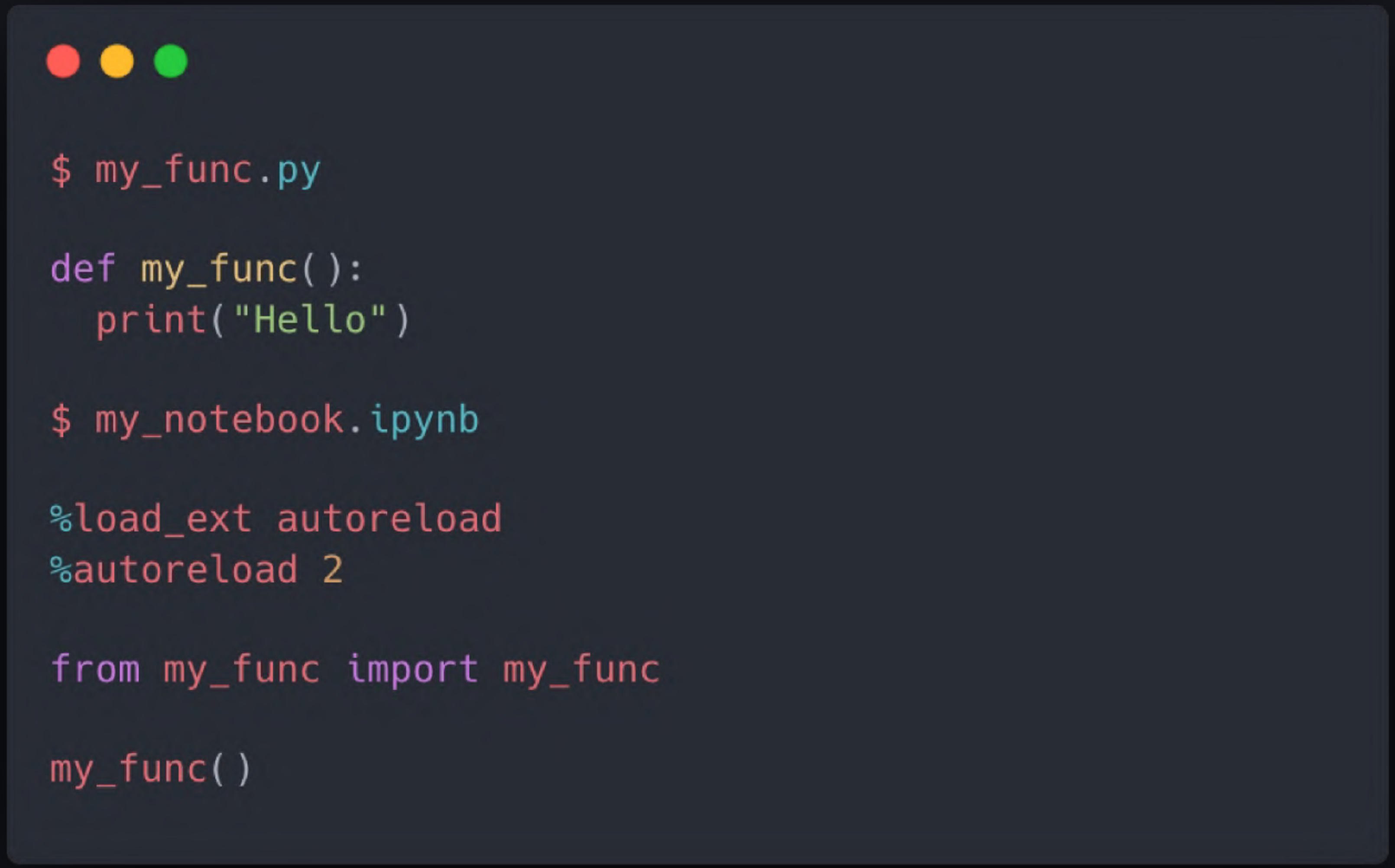
👉 Working with notebooks

Data

✅ Working with notebooks

👉 Extract functions to files & test them

Extract functions to files & test them

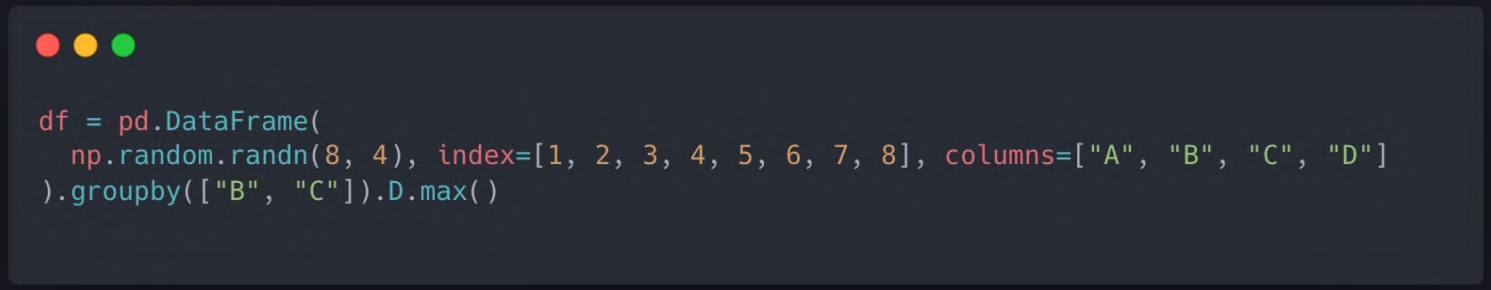


```
$ my_func.py

def my_func():
  print("Hello")

$ my_notebook.ipynb

%load_ext autoreload
%autoreload 2

from my_func import my_func

my_func()
```

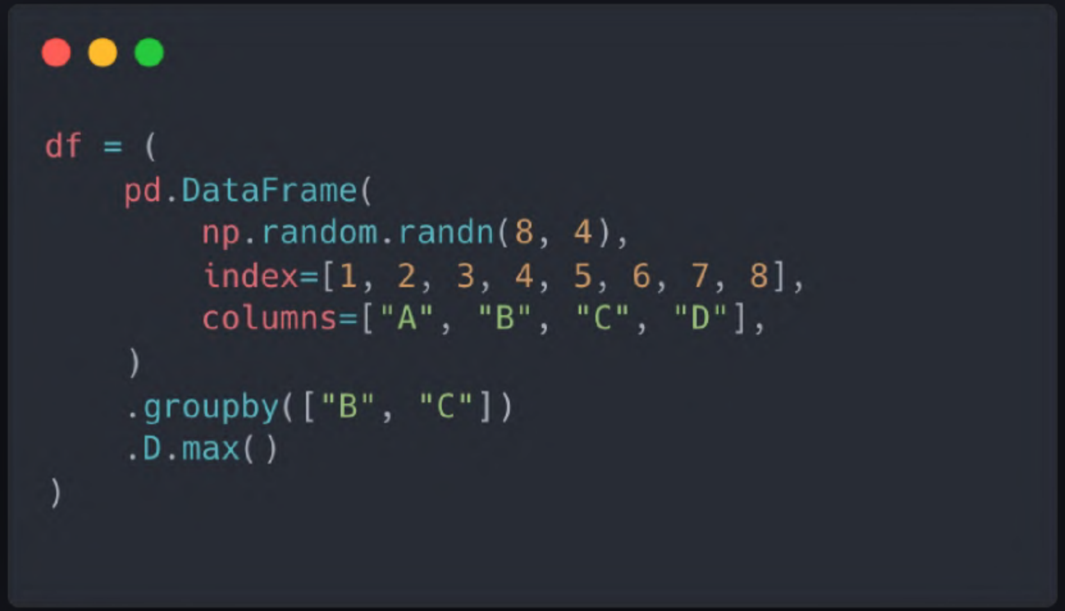
Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
- 👉 Autoformat your notebook code

Autoformat your notebook code

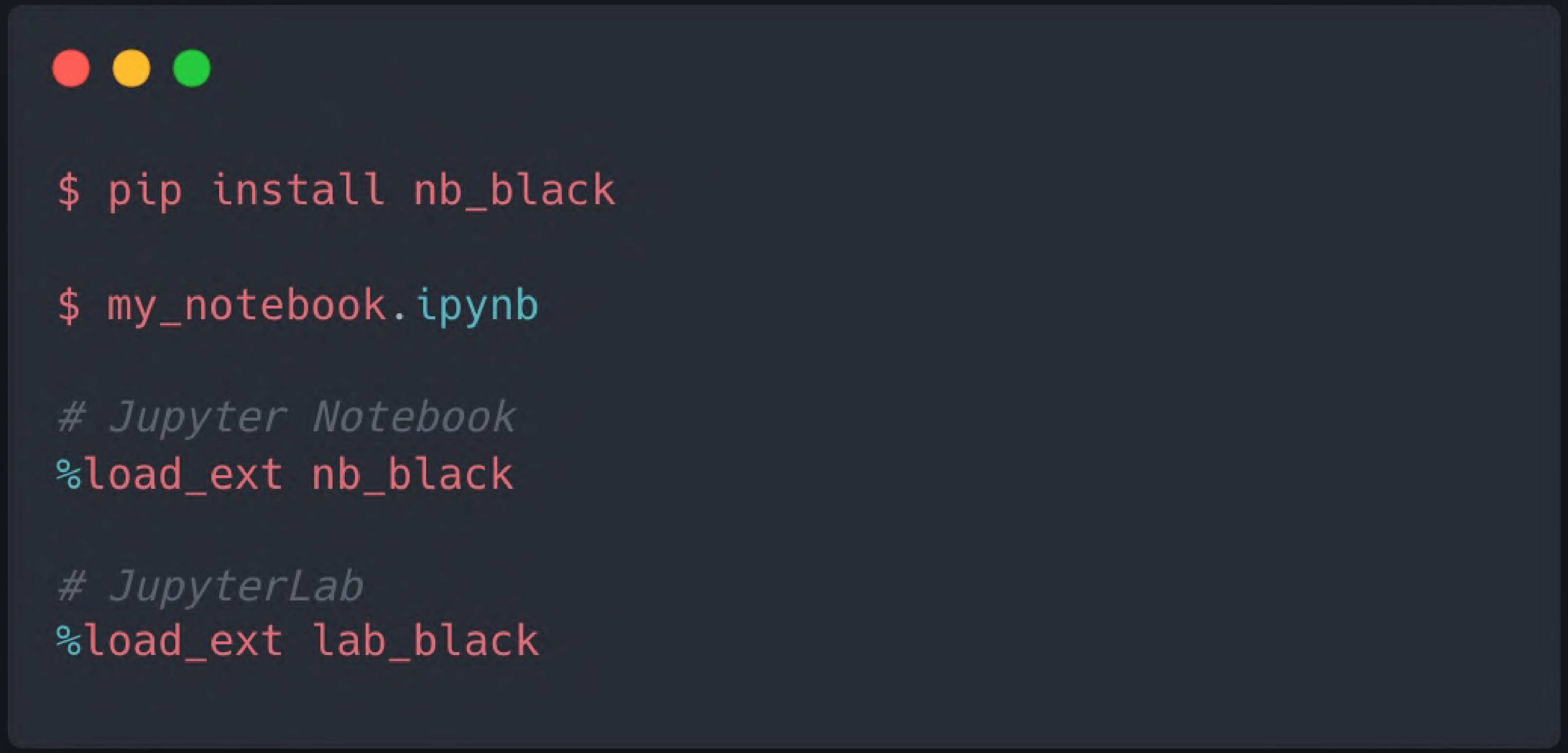


```
df = pd.DataFrame(
  np.random.randn(8, 4), index=[1, 2, 3, 4, 5, 6, 7, 8], columns=["A", "B", "C", "D"]
).groupby(["B", "C"]).D.max()
```



```
df = (
    pd.DataFrame(
        np.random.randn(8, 4),
        index=[1, 2, 3, 4, 5, 6, 7, 8],
        columns=["A", "B", "C", "D"],
    )
    .groupby(["B", "C"])
    .D.max()
)
```

Autoformat your notebook code



```
$ pip install nb_black

$ my_notebook.ipynb

# Jupyter Notebook
%load_ext nb_black

# JupyterLab
%load_ext lab_black
```

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
- 👉 Make your notebooks reproducible

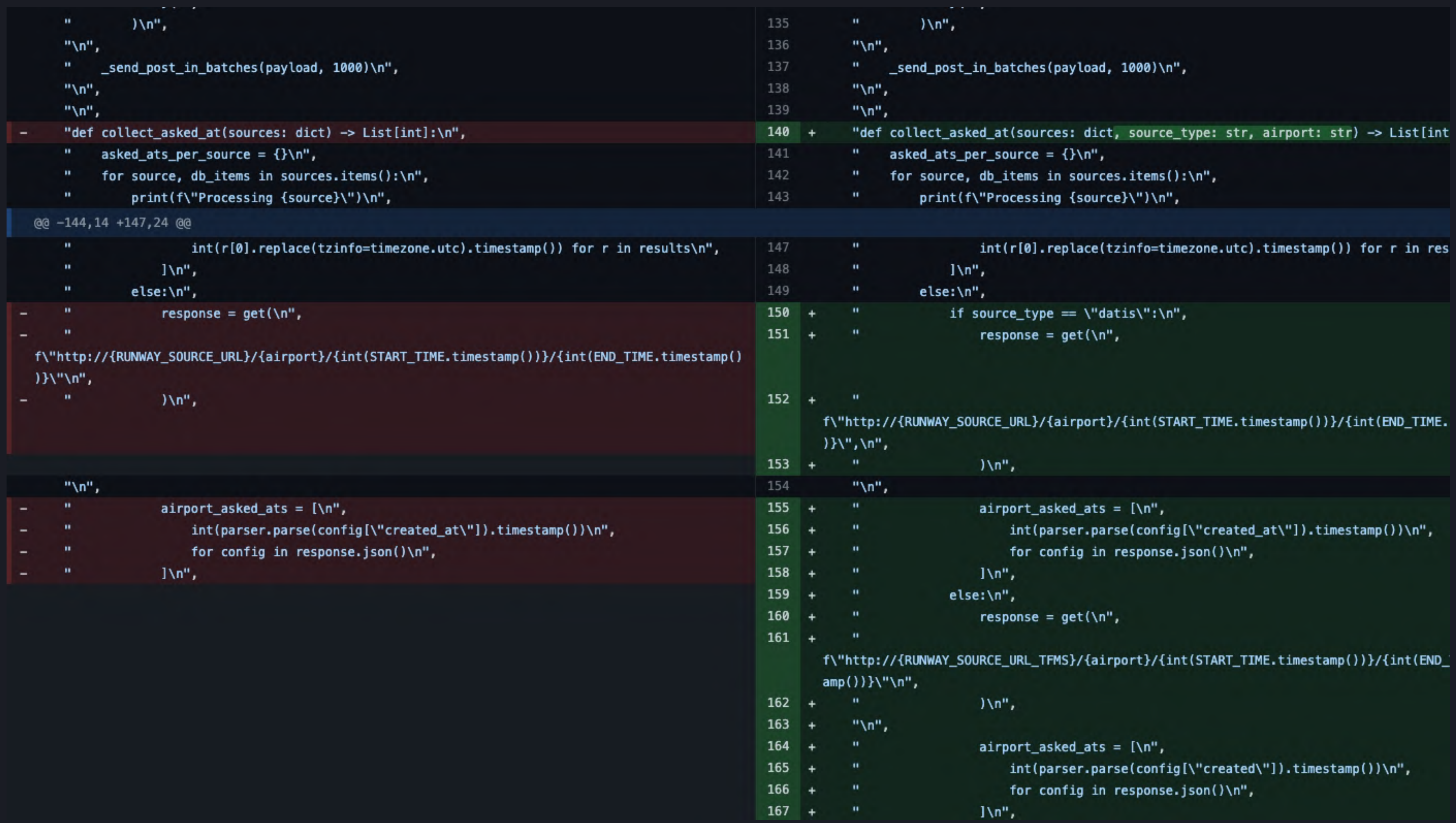
Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
- 👉 Clear the notebook outputs

Data

- ✅ Working with notebooks
    - ✅ Extract functions to files & test them
    - ✅ Autoformat your notebook code
    - ✅ Make your notebooks reproducible
    - ✅ Clear the notebook outputs
- 👉 Review notebook code

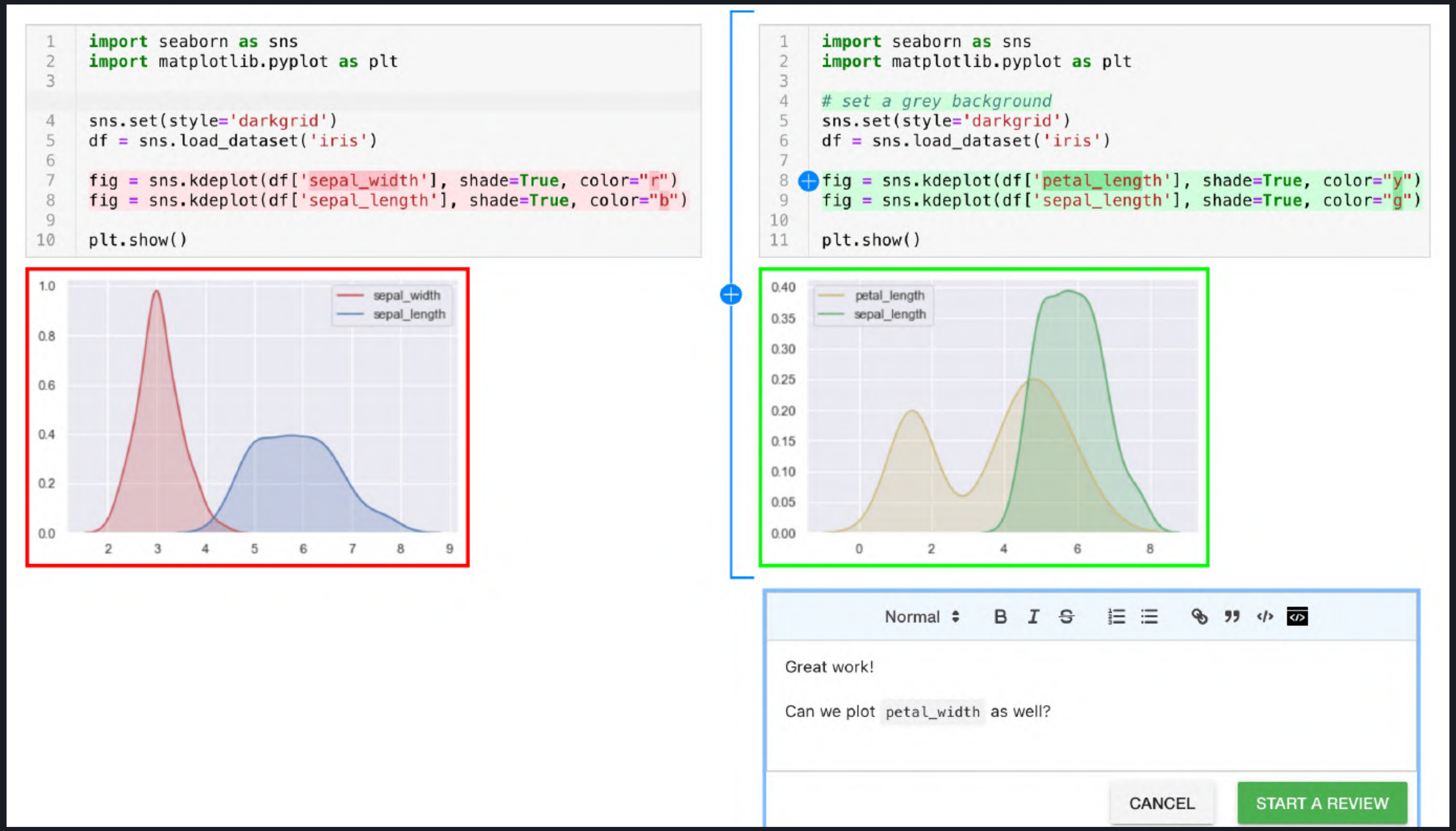
Review notebook code GitHub



```
   "        )\n",
   "\n",
   "    _send_post_in_batches(payload, 1000)\n",
   "\n",
   "\n",
-  "def collect_asked_at(sources: dict) -> List[int]:\n",
   "    asked_ats_per_source = {}\n",
   "    for source, db_items in sources.items():\n",
   "        print(f\"Processing {source}\")\n",
@@ -144,14 +147,24 @@
   "                int(r[0].replace(tzinfo=timezone.utc).timestamp()) for r in results\n",
   "            ]\n",
   "        else:\n",
-  "            response = get(\n",
-  "
f\"http://{RUNWAY_SOURCE_URL}/{airport}/{int(START_TIME.timestamp())}/{int(END_TIME.timestamp()
)}\"\n",
-  "            )\n",
   "\n",
-  "            airport_asked_ats = [\n",
-  "                int(parser.parse(config[\"created_at\"]).timestamp())\n",
-  "                for config in response.json()\n",
-  "            ]\n",
```

```
   "        )\n",
   "\n",
   "    _send_post_in_batches(payload, 1000)\n",
   "\n",
   "\n",
+  "def collect_asked_at(sources: dict, source_type: str, airport: str) -> List[int
   "    asked_ats_per_source = {}\n",
   "    for source, db_items in sources.items():\n",
   "        print(f\"Processing {source}\")\n",

   "                int(r[0].replace(tzinfo=timezone.utc).timestamp()) for r in res
   "            ]\n",
   "        else:\n",
+  "            if source_type == \"datis\":\n",
+  "                response = get(\n",
+  "
f\"http://{RUNWAY_SOURCE_URL}/{airport}/{int(START_TIME.timestamp())}/{int(END_TIME.
)}\",\n",
+  "                )\n",
   "\n",
+  "                airport_asked_ats = [\n",
+  "                    int(parser.parse(config[\"created_at\"]).timestamp())\n",
+  "                    for config in response.json()\n",
+  "                ]\n",
+  "            else:\n",
+  "                response = get(\n",
+  "
f\"http://{RUNWAY_SOURCE_URL_TFMS}/{airport}/{int(START_TIME.timestamp())}/{int(END_
amp())}\"\n",
+  "                )\n",
+  "\n",
+  "                airport_asked_ats = [\n",
+  "                    int(parser.parse(config[\"created\"]).timestamp())\n",
+  "                    for config in response.json()\n",
+  "                ]\n",
```

Review notebook code
ReviewNB



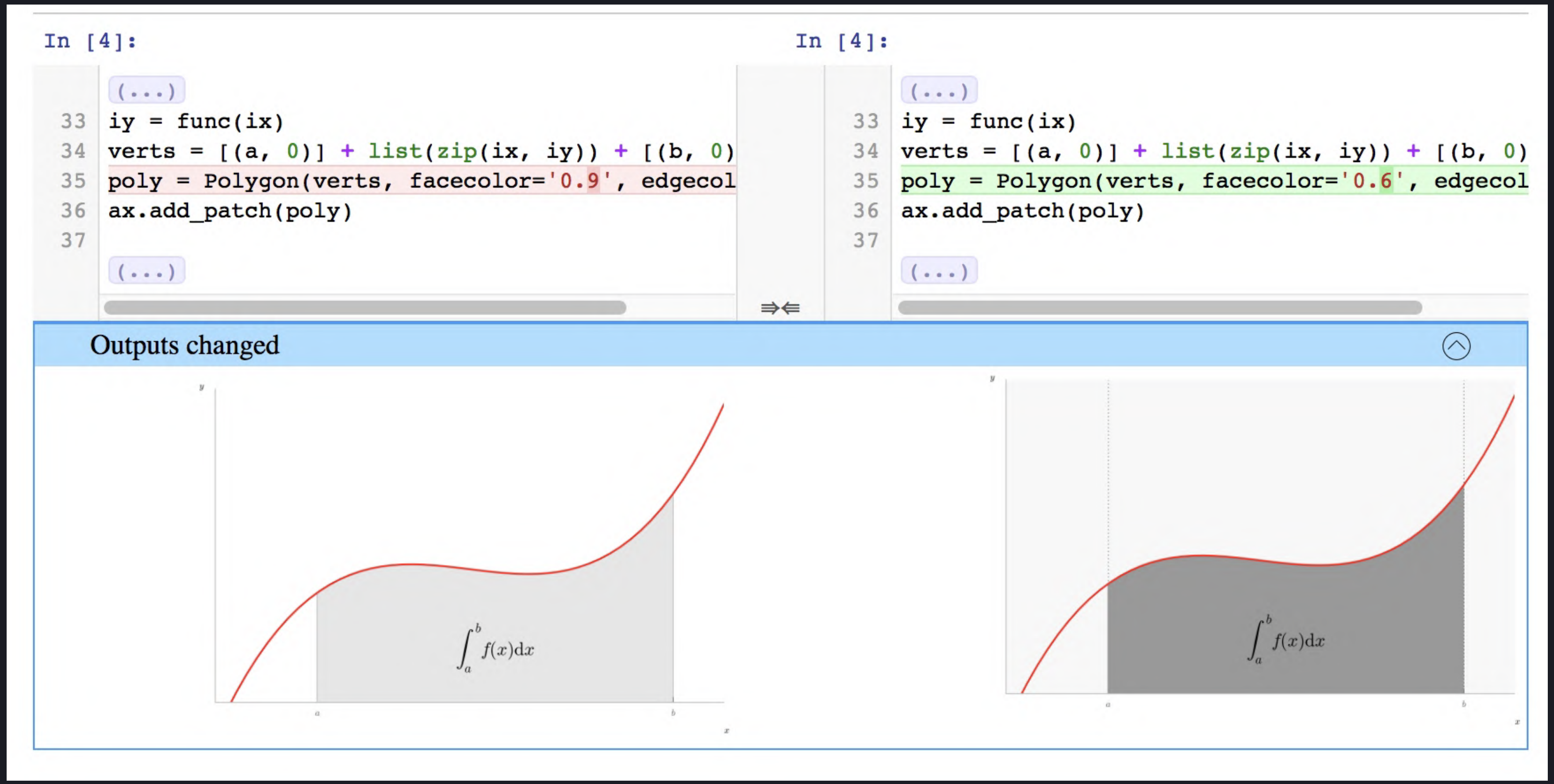
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style='darkgrid')
df = sns.load_dataset('iris')

fig = sns.kdeplot(df['sepal_width'], shade=True, color="r")
fig = sns.kdeplot(df['sepal_length'], shade=True, color="b")

plt.show()
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# set a grey background
sns.set(style='darkgrid')
df = sns.load_dataset('iris')

fig = sns.kdeplot(df['petal_length'], shade=True, color="y")
fig = sns.kdeplot(df['sepal_length'], shade=True, color="g")

plt.show()
```

Great work!

Can we plot `petal_width` as well?

Review notebook code
nbdime



```
In [4]:
(...)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.9', edgecol
ax.add_patch(poly)

(...)
```

```
In [4]:
(...)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.6', edgecol
ax.add_patch(poly)

(...)
```

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
  - ✅ Clear the notebook outputs
  - ✅ Review notebook code

👉 Sharing data processing code

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
  - ✅ Clear the notebook outputs
  - ✅ Review notebook code

- ✅ Sharing data processing code
  - 👉 Reuse functions for cleaning/processing from notebooks

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
  - ✅ Clear the notebook outputs
  - ✅ Review notebook code

- ✅ Sharing data processing code
  - ✅ Reuse functions for cleaning/processing from notebooks
  - 👉 Use same code in batch & streaming processing

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
  - ✅ Clear the notebook outputs
  - ✅ Review notebook code

- ✅ Sharing data processing code
  - ✅ Reuse functions for cleaning/processing from notebooks
  - ✅ Use same code in batch & streaming processing
- 👉 Share your pipelines between training & inference

Data

- ✅ Working with notebooks
  - ✅ Extract functions to files & test them
  - ✅ Autoformat your notebook code
  - ✅ Make your notebooks reproducible
  - ✅ Clear the notebook outputs
  - ✅ Review notebook code

- ✅ Sharing data processing code
  - ✅ Reuse functions for cleaning/processing from notebooks
  - ✅ Use same code in batch & streaming processing
  - ✅ Share your pipelines between training & inference
  - 👉 Be careful with dropping data

Data

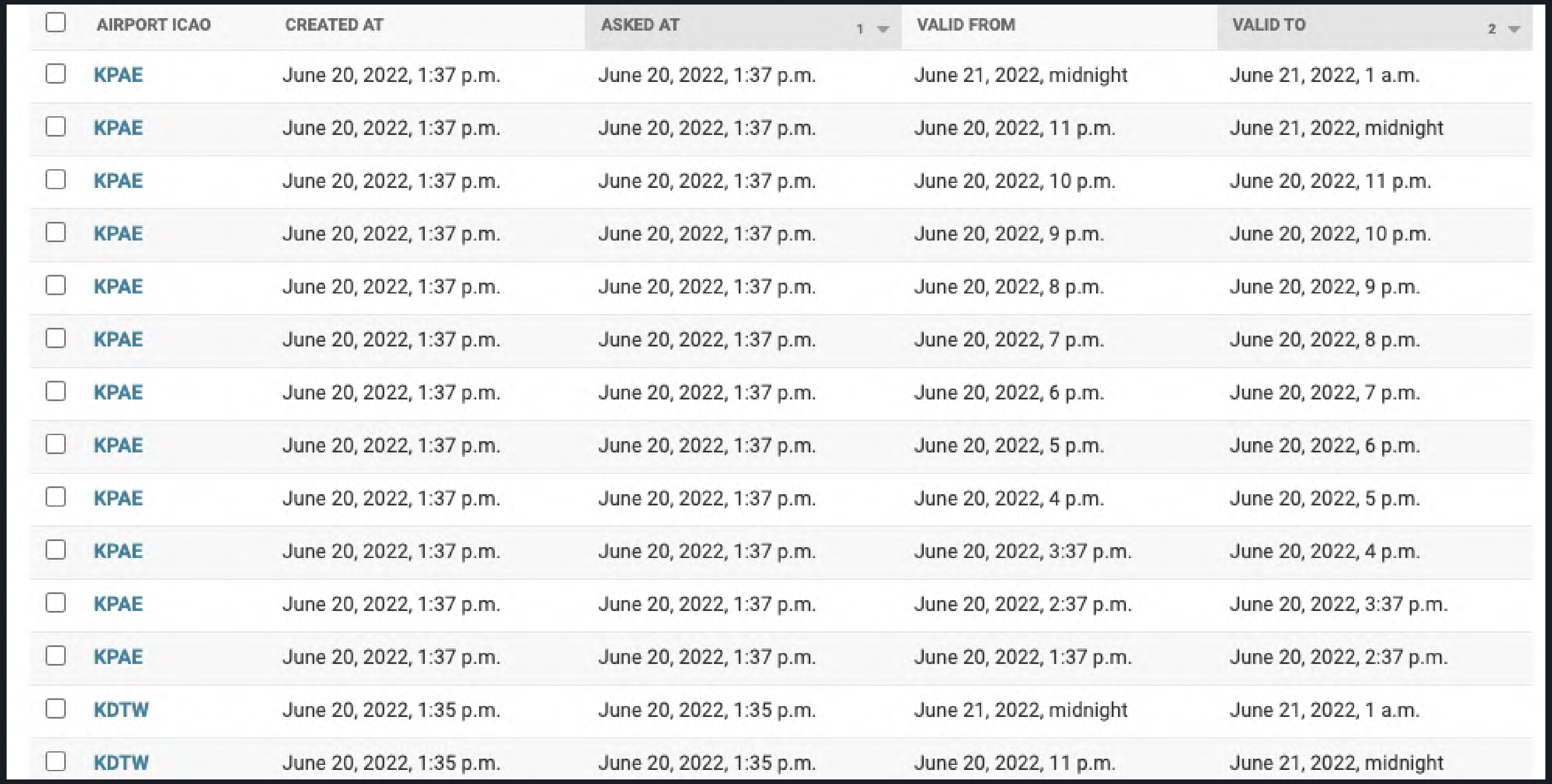
👉 Versioning data

Data

✅ Versioning data

👉 Store results of processing with created_at timestamp

Store results of processing with created_at timestamp



AIRPORT ICAO	CREATED AT	ASKED AT 1	VALID FROM	VALID TO 2
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 21, 2022, midnight	June 21, 2022, 1 a.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 11 p.m.	June 21, 2022, midnight
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 10 p.m.	June 20, 2022, 11 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 9 p.m.	June 20, 2022, 10 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 8 p.m.	June 20, 2022, 9 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 7 p.m.	June 20, 2022, 8 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 6 p.m.	June 20, 2022, 7 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 5 p.m.	June 20, 2022, 6 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 4 p.m.	June 20, 2022, 5 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 3:37 p.m.	June 20, 2022, 4 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 2:37 p.m.	June 20, 2022, 3:37 p.m.
KPAE	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 1:37 p.m.	June 20, 2022, 2:37 p.m.
KDTW	June 20, 2022, 1:35 p.m.	June 20, 2022, 1:35 p.m.	June 21, 2022, midnight	June 21, 2022, 1 a.m.
KDTW	June 20, 2022, 1:35 p.m.	June 20, 2022, 1:35 p.m.	June 20, 2022, 11 p.m.	June 21, 2022, midnight

Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp

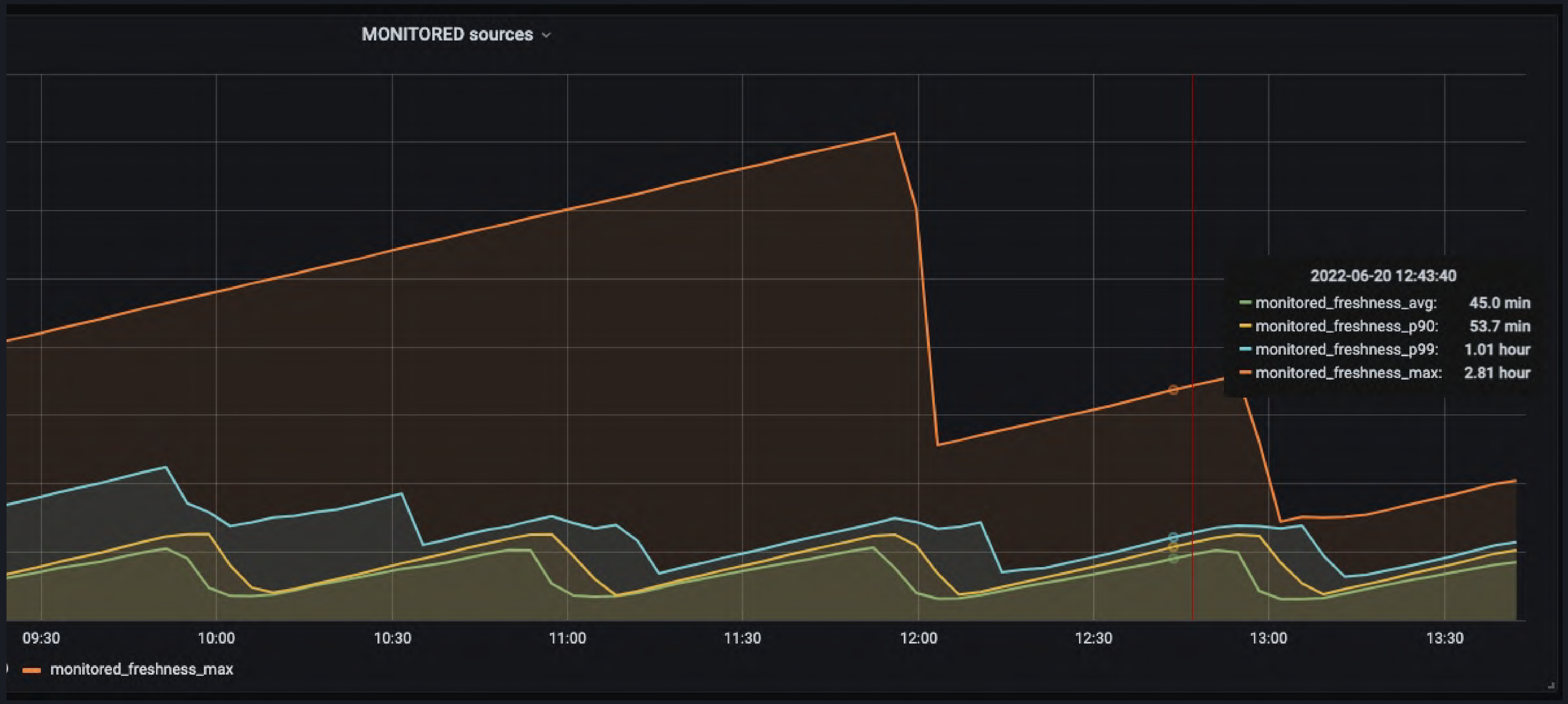
👉 Monitoring data quality

Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp

- ✅ Monitoring data quality
  - 👉 Data freshness

Data freshness



Data

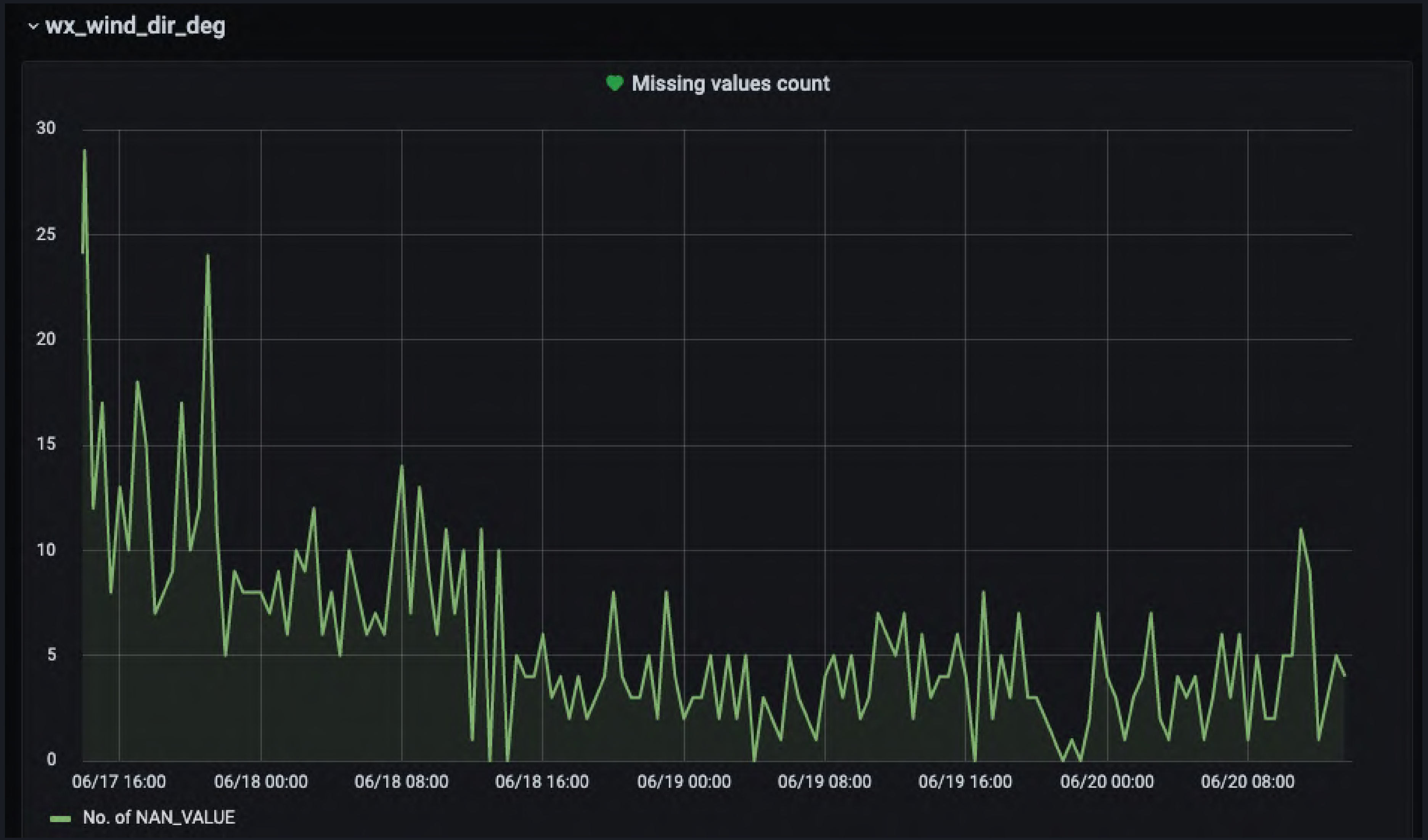
- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp
- ✅ Monitoring data quality
  - ✅ Data freshness
  - 👉 Dropped samples

Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp

- ✅ Monitoring data quality
  - ✅ Data freshness
  - ✅ Dropped samples
  - 👉 Missing values

Missing values

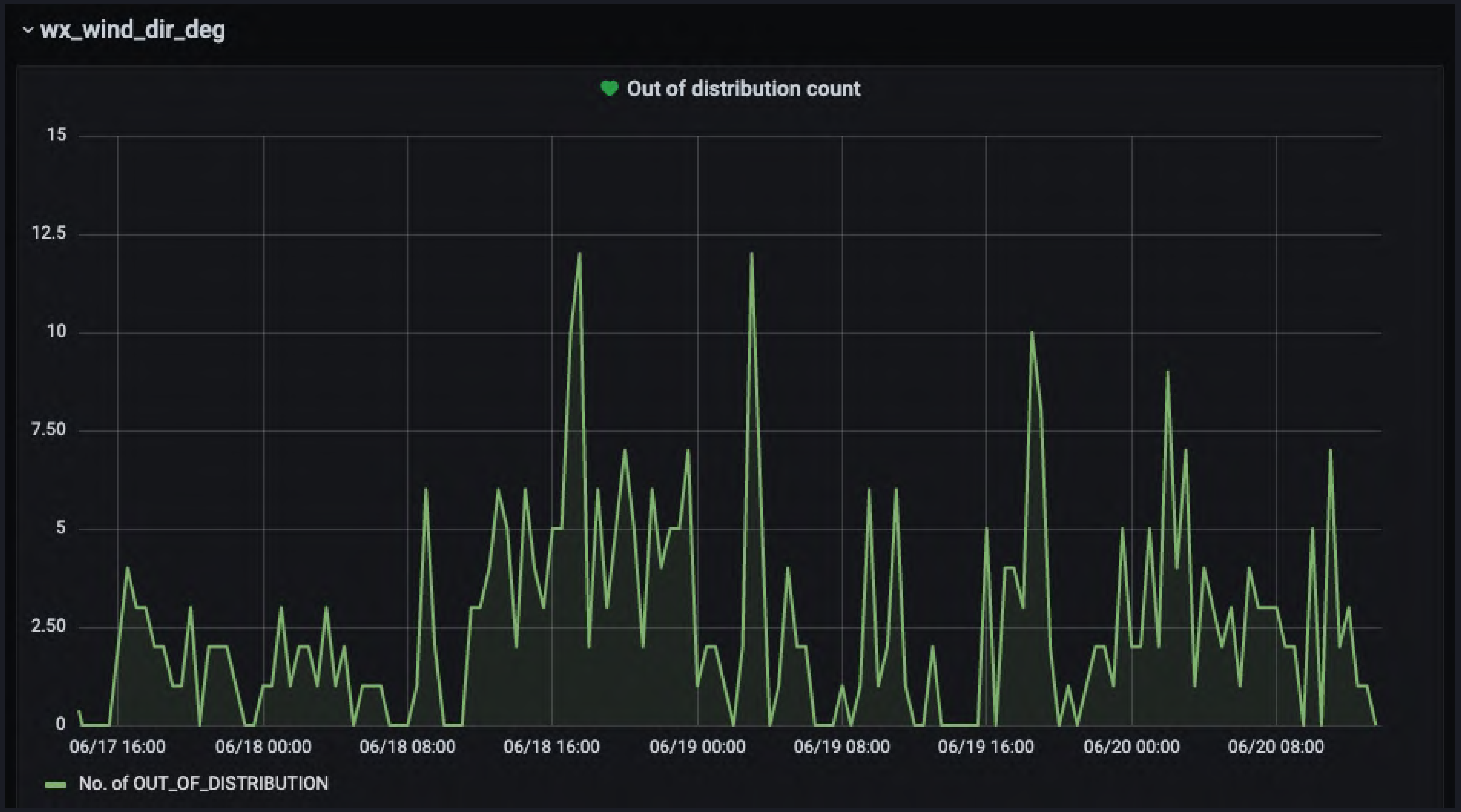


Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp

- ✅ Monitoring data quality
  - ✅ Data freshness
  - ✅ Dropped samples
  - ✅ Missing values
  - 👉 Out-of-distribution values

Out-of-distribution values



Data

- ✅ Versioning data
  - ✅ Store results of processing
    with created_at timestamp

- ✅ Monitoring data quality
  - ✅ Data freshness
  - ✅ Dropped samples
  - ✅ Missing values
  - ✅ Out-of-distribution values
  - 👉 Zeros

Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp

- ✅ Monitoring data quality
  - ✅ Data freshness
  - ✅ Dropped samples
  - ✅ Missing values
  - ✅ Out-of-distribution values
  - ✅ Zeros
  - 👉 General distribution

Data

- ✅ Versioning data
  - ✅ Store results of processing with created_at timestamp
- ✅ Monitoring data quality
  - ✅ Data freshness
  - ✅ Dropped samples
  - ✅ Missing values
  - ✅ Out-of-distribution values
  - ✅ Zeros
  - ✅ General distribution

👉 Documenting data

Models

Models

👉 Versioning models & pipelines

Models

✅ Versioning models & pipelines

👉 Write pipeline compliant training code

Models

- ✅ Versioning models & pipelines
  - ✅ Write pipeline compliant training code
- 👉 Track experiment versions in UI

Models

- ✅ Versioning models & pipelines
  - ✅ Write pipeline compliant training code
  - ✅ Track experiment versions in UI
- 👉 Add ability to run old experiments

Models

- ✅ Versioning models & pipelines
  - ✅ Write pipeline compliant training code
  - ✅ Track experiment versions in UI
  - ✅ Add ability to run old experiments
  - 👉 Allow dynamic feature sets

Models

- ✅ Versioning models & pipelines
  - ✅ Write pipeline compliant training code
  - ✅ Track experiment versions in UI
  - ✅ Add ability to run old experiments
  - ✅ Allow dynamic feature sets
- 👉 Version models with their pipeline version

Models

- ✅ Versioning models & pipelines
  - ✅ Write pipeline compliant training code
  - ✅ Track experiment versions in UI
  - ✅ Add ability to run old experiments
  - ✅ Allow dynamic feature sets
  - ✅ Version models with their pipeline version

👉 Separate & scale

feature generation

models deployment

Models

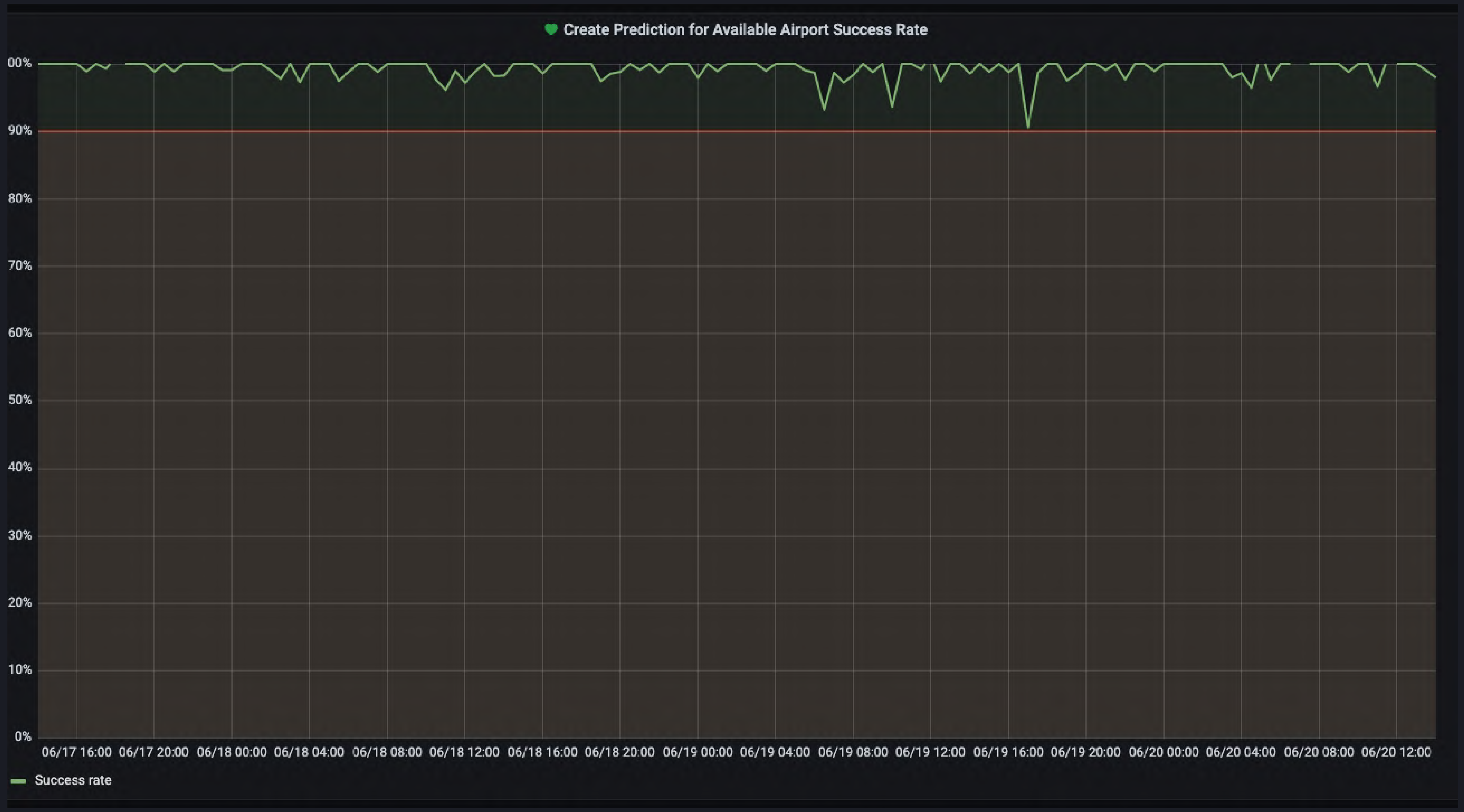
👉 Monitoring models

Models

✅ Monitoring models

👉 Success rate of generating a prediction

Success rate of generating a prediction



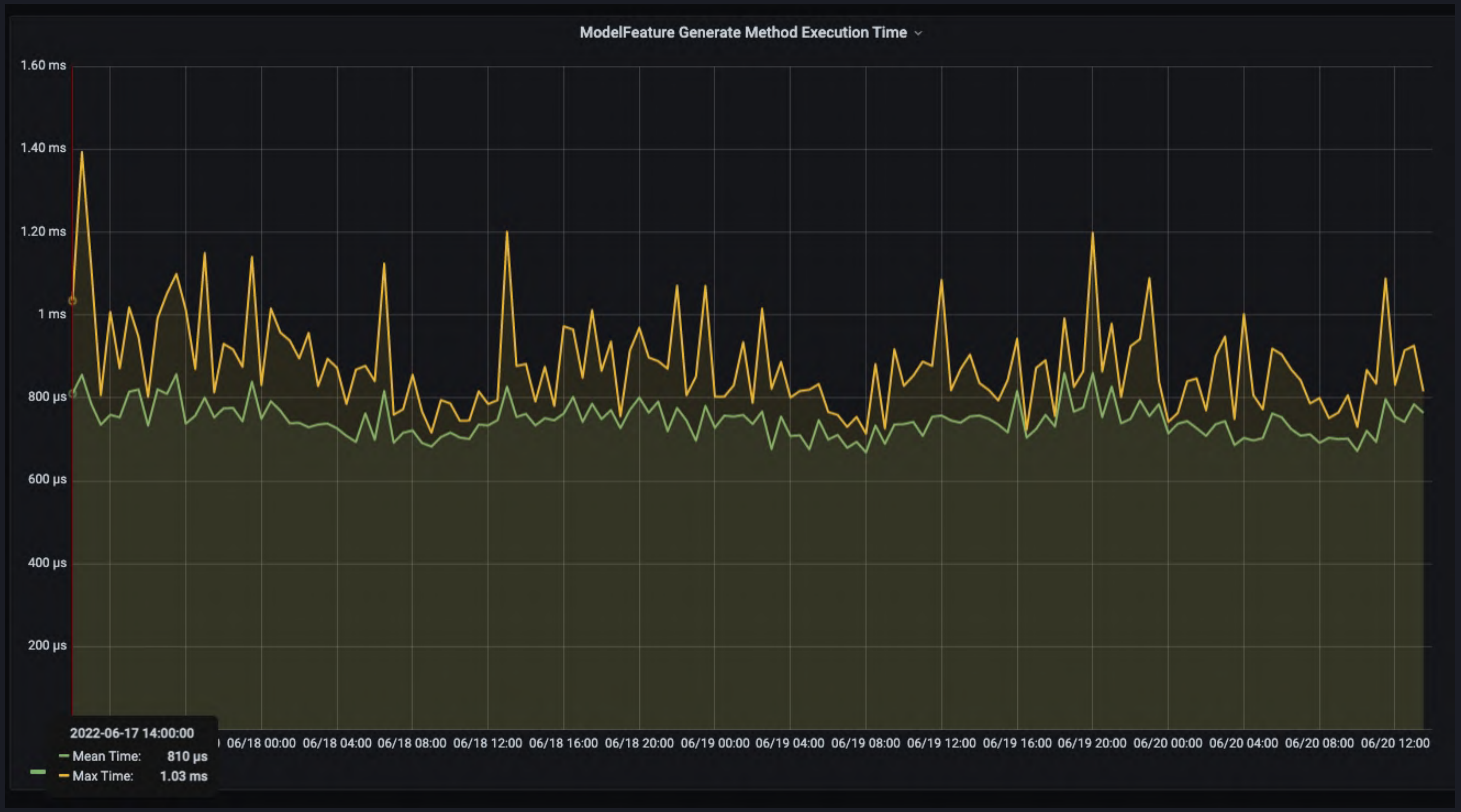
Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
- 👉 Model performance vs. non-ML baseline

Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
  - ✅ Model performance vs. non-ML baseline
- 👉 Feature generation/fetching time

Feature generation/fetching time



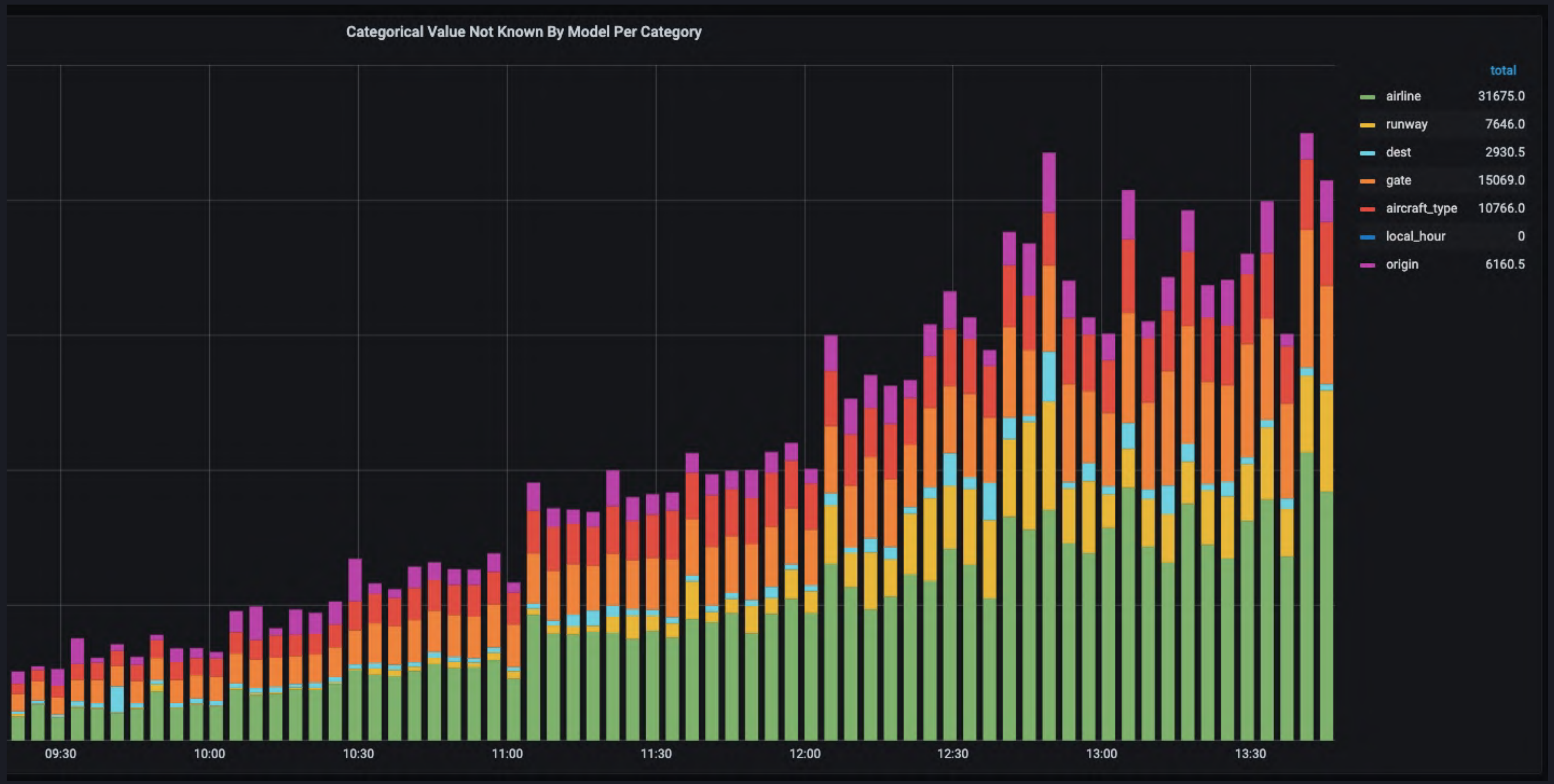
Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
  - ✅ Model performance vs. non-ML baseline
  - ✅ Feature generation/fetching time
- 👉 Silent failures

Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
  - ✅ Model performance vs. non-ML baseline
  - ✅ Feature generation/fetching time
  - ✅ Silent failures
- 👉 Unknown categorical values

Unknown categorical values



Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
  - ✅ Model performance vs. non-ML baseline
  - ✅ Feature generation/fetching time
  - ✅ Silent failures
  - ✅ Unknown categorical values
- 👉 Monitor output like you monitor data

Models

- ✅ Monitoring models
  - ✅ Success rate of generating a prediction
  - ✅ Model performance vs. non-ML baseline
  - ✅ Feature generation/fetching time
  - ✅ Silent failures
  - ✅ Unknown categorical values
  - ✅ Monitor output like you monitor data
- 👉 Sanity check output

Iteration

Iteration

👉 Start simple & get to working solution

Iteration

✅ Start simple & get to working solution

👉 Make sure the data & model pipeline are correct

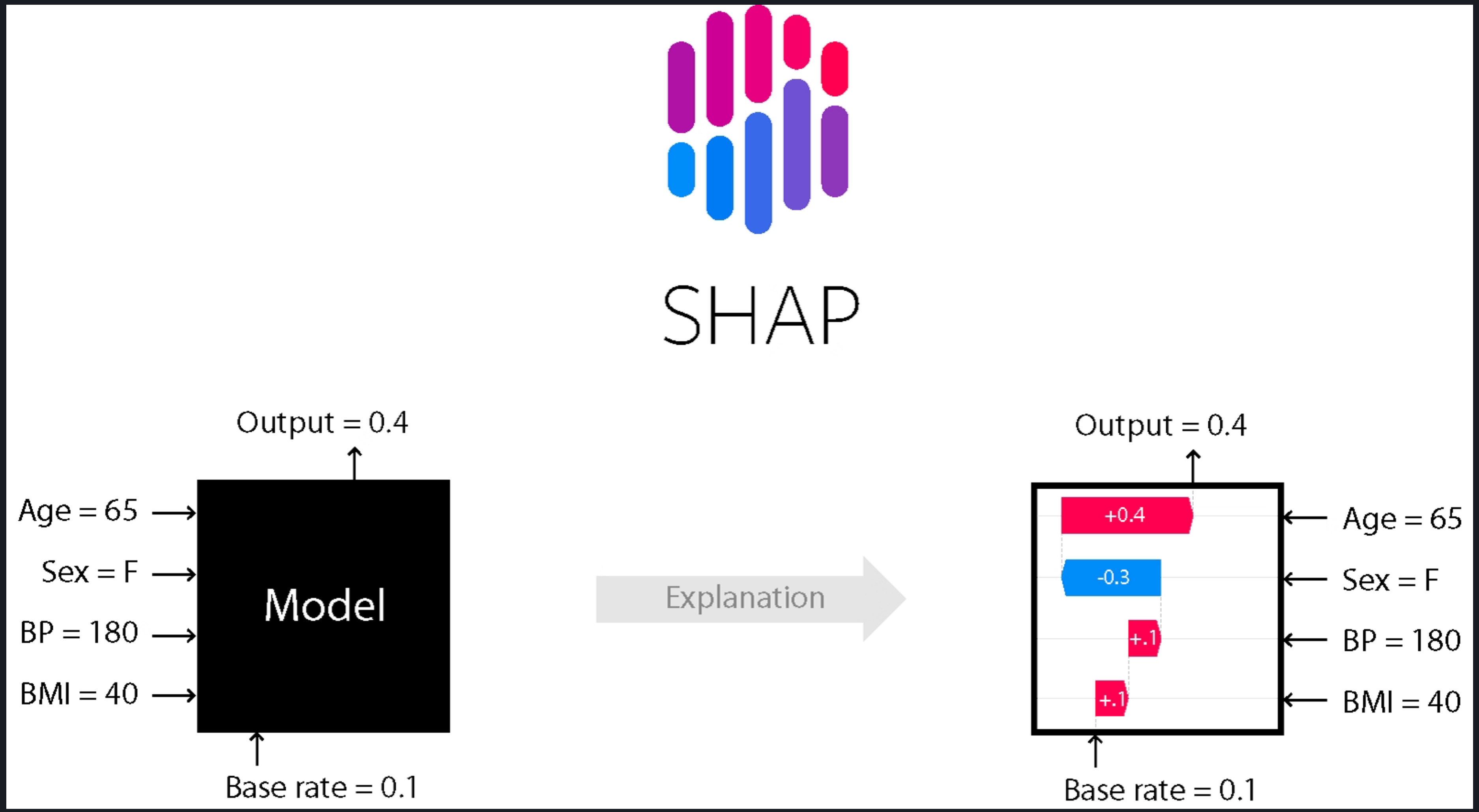
Iteration

✅ Start simple & get to working solution

✅ Make sure the data & model pipeline are correct

👉 Explain the models

Explain the models



Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- 👉 Reuse hyperparameters

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- 👉 Plan work based on performance improvements potential

Iteration

✅ Start simple & get to working solution
✅ Make sure the data & model pipeline
are correct
✅ Explain the models
✅ Reuse hyperparameters
✅ Plan work based on
performance improvements potential
👉 Bugs/unhandled cases

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- ✅ Plan work based on performance improvements potential
  - ✅ Bugs/unhandled cases
  - 👉 New features

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- ✅ Plan work based on performance improvements potential
  - ✅ Bugs/unhandled cases
  - ✅ New features
  - 👉 Model architecture

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- ✅ Plan work based on performance improvements potential
  - ✅ Bugs/unhandled cases
  - ✅ New features
  - ✅ Model architecture
  - 👉 More data

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- ✅ Plan work based on performance improvements potential
  - ✅ Bugs/unhandled cases
  - ✅ New features
  - ✅ Model architecture
  - ✅ More data

👉 Don't implement all practices at once

Iteration

- ✅ Start simple & get to working solution
- ✅ Make sure the data & model pipeline are correct
- ✅ Explain the models
- ✅ Reuse hyperparameters
- ✅ Plan work based on performance improvements potential
  - ✅ Bugs/unhandled cases
  - ✅ New features
  - ✅ Model architecture
  - ✅ More data

- ✅ Don't implement all practices at once

👉 Knowledge sharing

AIRSPACE
INTELLIGENCE

PIOTR MAZUR
PIOTR@AIRSPACE-INTELLIGENCE.COM