When labels are available

MONITORING MACHINE LEARNING IN PYTHON

Maciej Balawejder
Data Scientist

Estimated vs realized performance

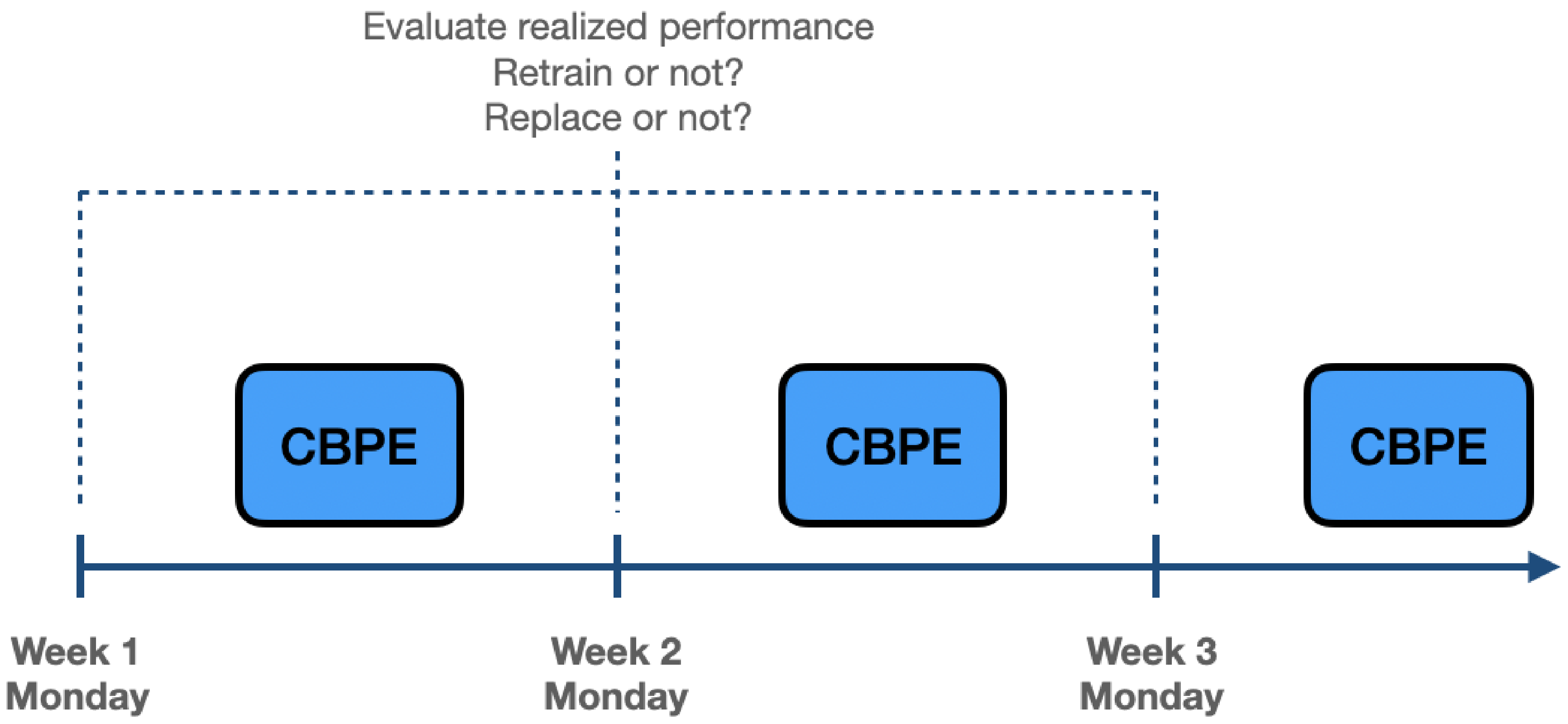
Estimated performance:

- measures how well model is expected to perform
- determined using **estimators** like CBPE, and DLE
- **estimated** when ground truth is not available

Realized performance:

- represents **measured** performance
- determined using performance **calculator**
- **calculated** when ground truth is available

Delayed ground truth



Performance calculator

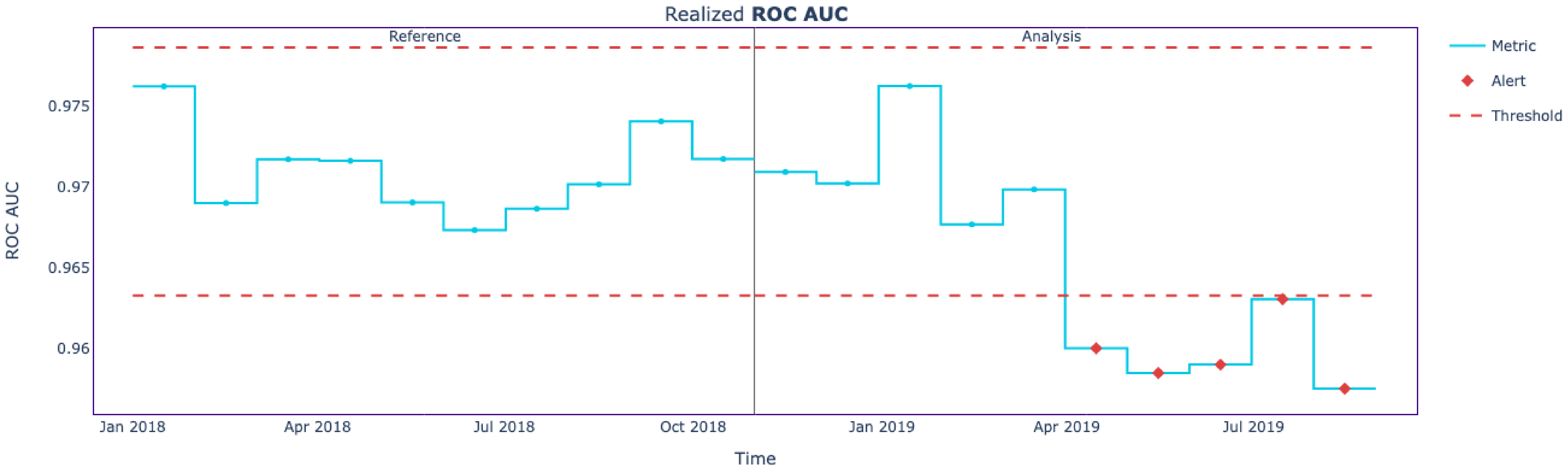
```python
# Intialize the calculator
calc = nannyml.PerformanceCalculator(
    y_pred_proba='y_pred_proba',
    y_pred='y_pred',
    y_true='arrived',
    timestamp_column_name='timestamp',
    problem_type='classification_binary',
    chunk_period='d',
    metrics=['roc_auc', 'accuracy'],
    )
```

```python
# Fit the calculator
calc.fit(reference)
realized_results = calc.calculate(analysis)
```

Plot the results

```
# Show realized performance plot
results.plot().show()
```

Realized performance

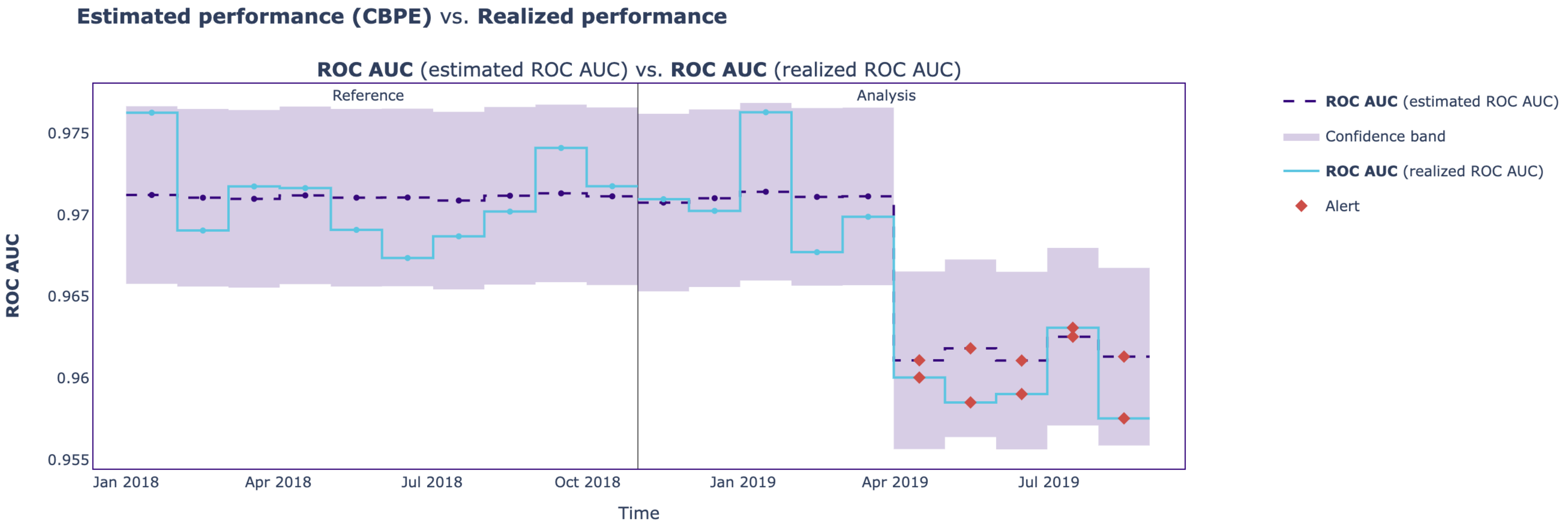


Realized and estimated performance

```
# Estimate and calculate results
estimated_results = estimator.estimate(analysis)
realized_results = calculator.calculate(analysis)

# Show comparison plot
realized_results.compare(estimated_results).plot().show()
```

Realized and estimated performance



Let's practice!

MONITORING MACHINE LEARNING IN PYTHON

datacamp

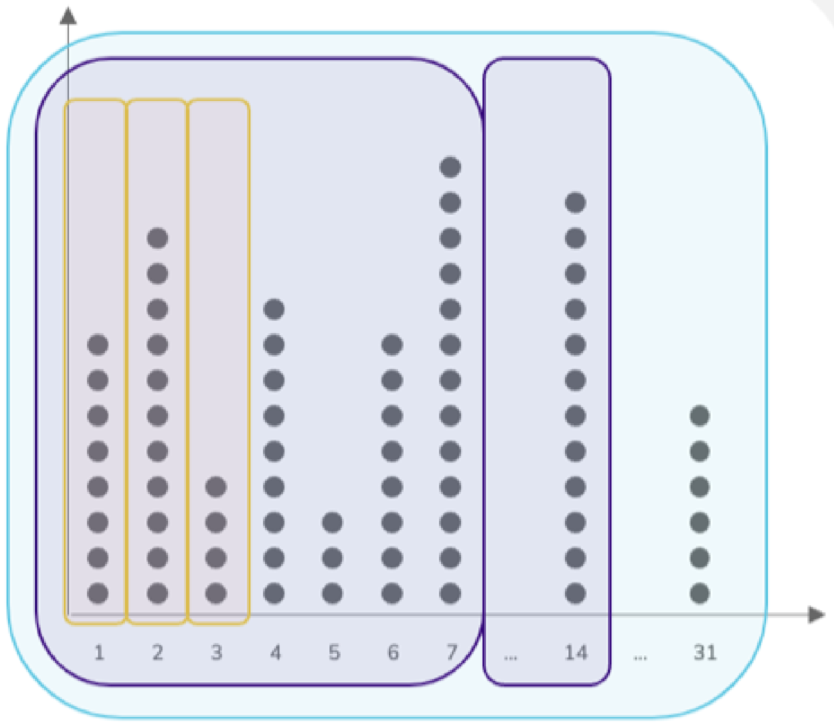
Working with calculated and estimated results

MONITORING MACHINE LEARNING IN PYTHON

Maciej Balawejder
Data Scientist

How to chunk the data?

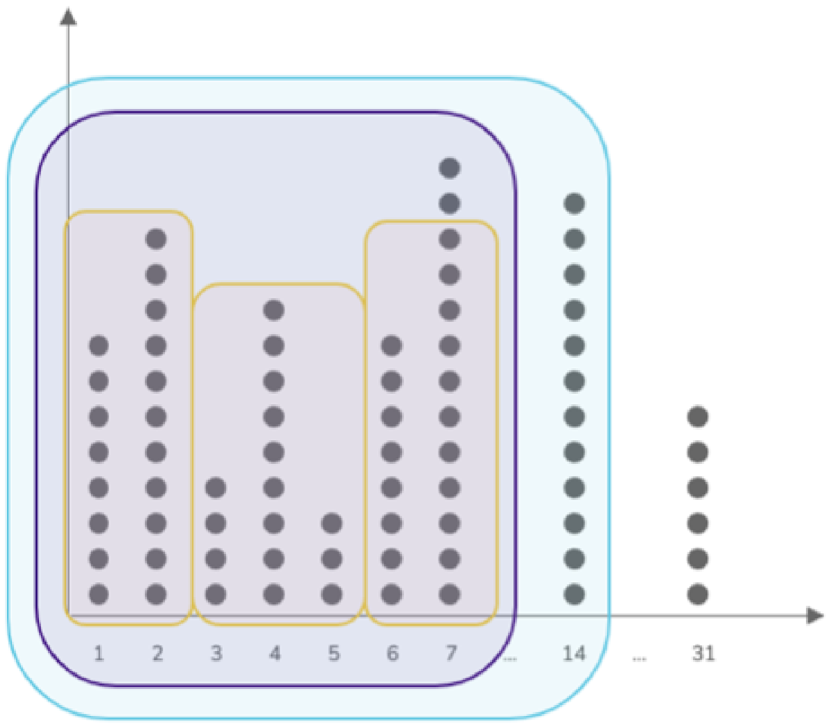
Time-based



Day

Daily – Weekly – Monthly

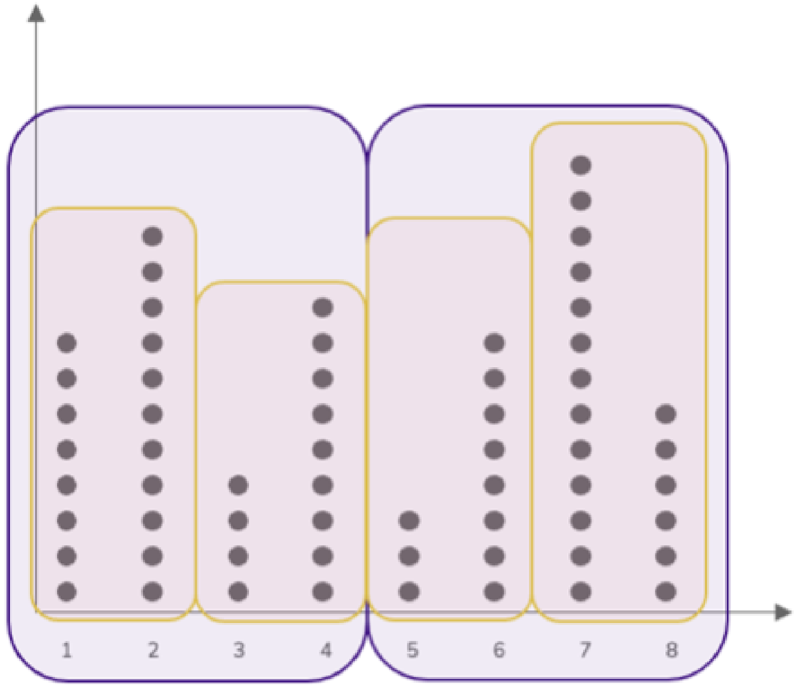
Size-based



Day

Every 15 – 45 – 90 points

Number-based



Total 4 – 2 number of chunks

Specifying different chunks

Chunking period arguments

Alias	Description
s	second
t	minute
h	hour
d	day
w	week
m	month
q	quarter
y	year

```
# Initialize the algorithm
cbpe = nannyml.CBPE(
    problem_type='classification_binary',
    y_pred_proba='predicted_probability',
    y_pred='prediction',
    y_true='employed',
    metrics=['roc_auc'],
    chunk_period='m',
    # chunk_size = 5000,
    # chunk_number = 10
)
```

Initializing custom thresholds

Standard deviation thresholds

- Manually set lower and upper standard deviation multiplier

```
# Standard deviation thresholds
stdt = StandardDeviationThreshold(
    std_lower_multiplier=3,
    std_upper_multiplier=3
    )
```

Constant thresholds

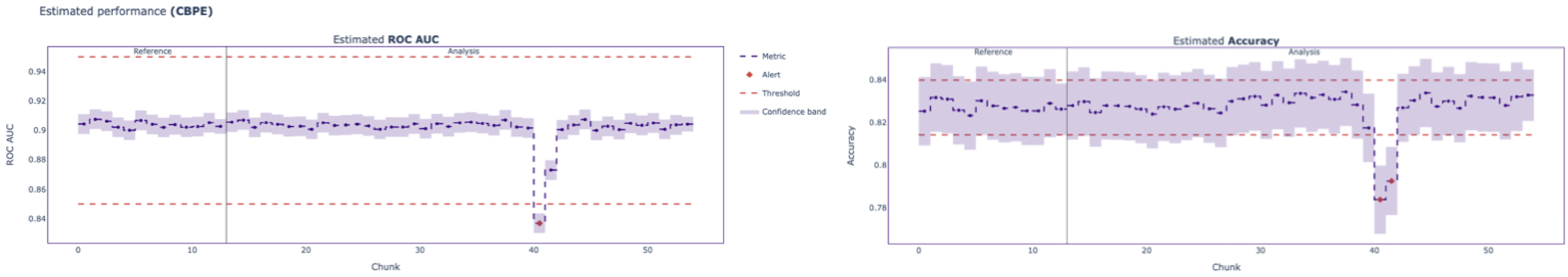
- Manually set the lower and upper threshold values

```
# Constant thresholds
ct = ConstantThreshold(
    lower=0.85,
    upper=0.95
    )
```

Specifying custom thresholds

```
# Import threshold methods(last slide)
from nannyml.thresholds import ConstantThreshold, StandardDeviationThreshold

# Passing thresholds to the CBPE algorithm
estimator = nannyml.CBPE(...
    metrics = ['roc_auc', 'accuracy'],
    thresholds={'roc_auc': ct, 'accuracy' : stdt}
)
```



Filtering results

- By period

```
filtered_results = results.filter(period='analysis')
```

- By metrics

```
filtered_results = results.filter(metrics=['mae'])
```

- Both

```
filtered_results = results.filter(period='analysis', metrics=['mae'])
```

Export results to dataframe

```
# Export results to dataframe format
results.filter(period='analysis').to_df()
```

	chunk							roc_auc							
	key	chunk_index	start_index	end_index	start_date	end_date	period	value	sampling_error	realized	upper_confidence_boundary	lower_confidence_boundary	upper_threshold	lower_threshold	alert
0	[0:4999]	0	0	4999	None	None	analysis	0.905547	0.002230	NaN	0.912236	0.898859	0.95	0.85	False
1	[5000:9999]	1	5000	9999	None	None	analysis	0.907030	0.002230	NaN	0.913719	0.900342	0.95	0.85	False
2	[10000:14999]	2	10000	14999	None	None	analysis	0.902044	0.002230	NaN	0.908733	0.895355	0.95	0.85	False
3	[15000:19999]	3	15000	19999	None	None	analysis	0.905250	0.002230	NaN	0.911939	0.898562	0.95	0.85	False
4	[20000:24999]	4	20000	24999	None	None	analysis	0.904054	0.002230	NaN	0.910742	0.897365	0.95	0.85	False

Let's practice!

MONITORING MACHINE LEARNING IN PYTHON

Business value calculation and estimation

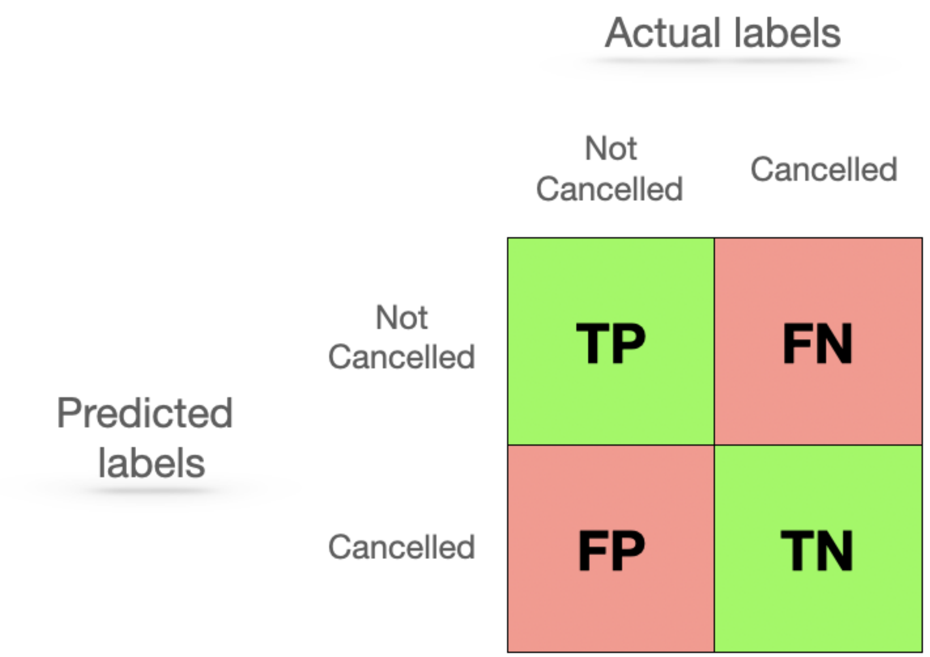
MONITORING MACHINE LEARNING IN PYTHON

Maciej Balawejder
Data Scientist

Model business value

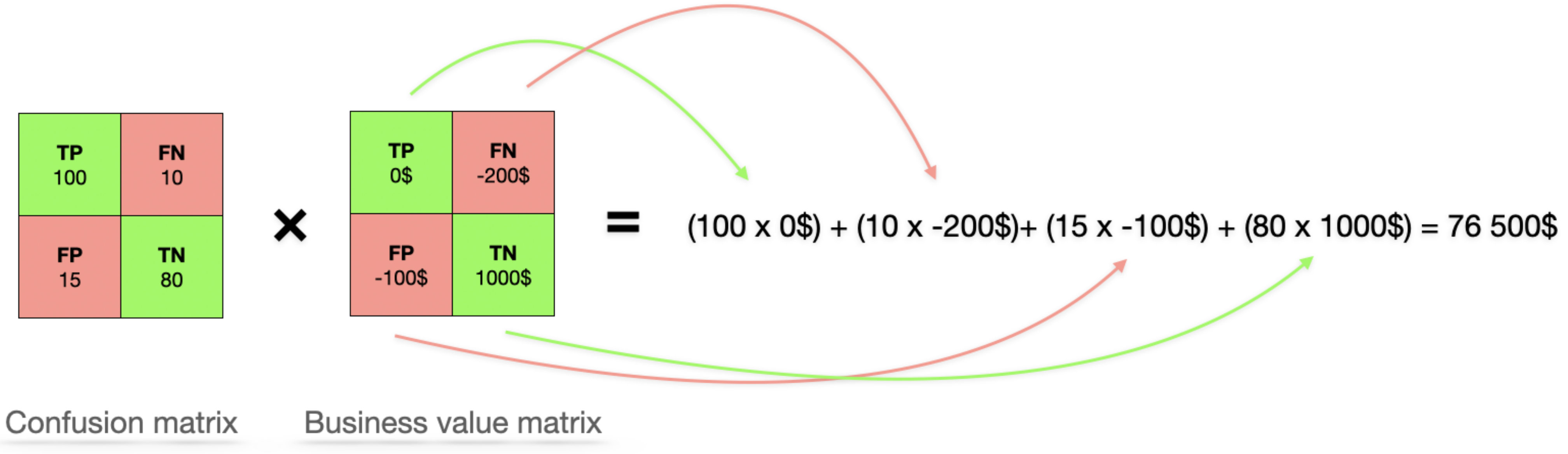
- The aim of machine learning model is to provide value to the business.
- The business value of the model can decrease due to:
  - Change in customer's habits
  - The model might not be useful anymore

Confusion matrix



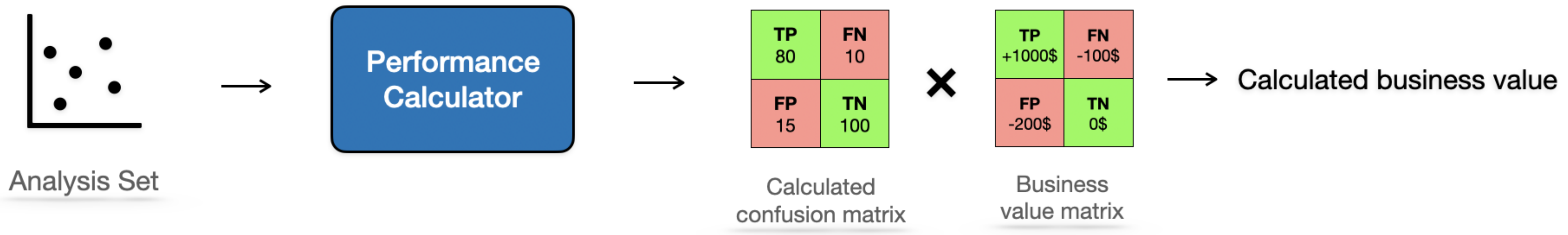
- True positive (TP) - the model correctly predicts that a booking will not be canceled
- False positive (FP) - the model incorrectly predicts a booking will not be canceled
- False negative (FN) - the model incorrectly predicts that a booking will be canceled
- True negative (TN) - the model correctly predicts that a booking will be canceled

Business value formula



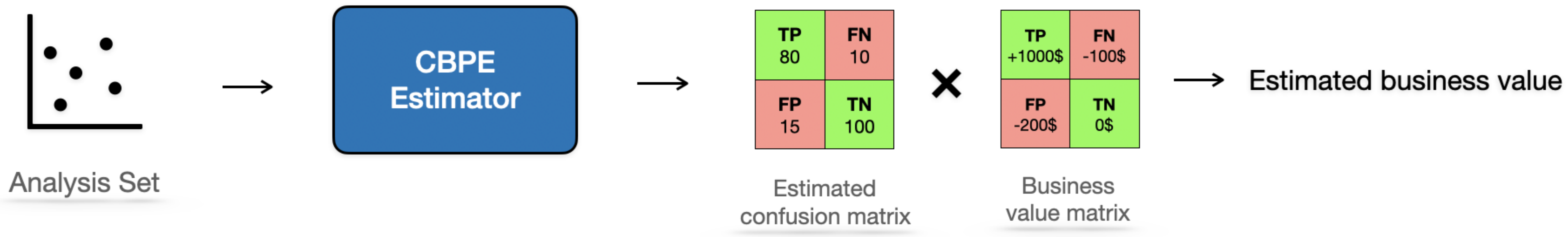
- True positive (TP) - doesn't add or subtract any value.
- False positive (FP) - leads to relocations and discounts, it costs hotel $200.
- False negative (FN) - costs hotel $100, a one-night stay until a replacement is found.
- True negative (TN) - worth $1000 because the hotel can rent the room to someone else.

When labels are available




```
# Initialize the calculator
calculator = nannyml.PerformanceCalculator(...
    problem_type='classification_binary',
    metrics=['business_value'],
    # [value_of_TN, value_of_FP], [value_of_FN, value_of_TP]]
    business_value_matrix = [[0, -200],[-100, 1000]],
    normalize_business_value='None')
```

When labels are not available




```python
# Initialize the estimator
estimator = nannyml.CBPE(...
    problem_type='classification_binary',
    metrics=['business_value'],
    # [value_of_TN, value_of_FP], [value_of_FN, value_of_TP]]
    business_value_matrix = [[0, -200],[-100, 1000]],
    normalize_business_value='per_prediction')
```

Let's practice!

MONITORING MACHINE LEARNING IN PYTHON