What is NannyML?

MONITORING MACHINE LEARNING IN PYTHON

Hakim Elakhrass
Co-founder and CEO of NannyML

Prerequisites

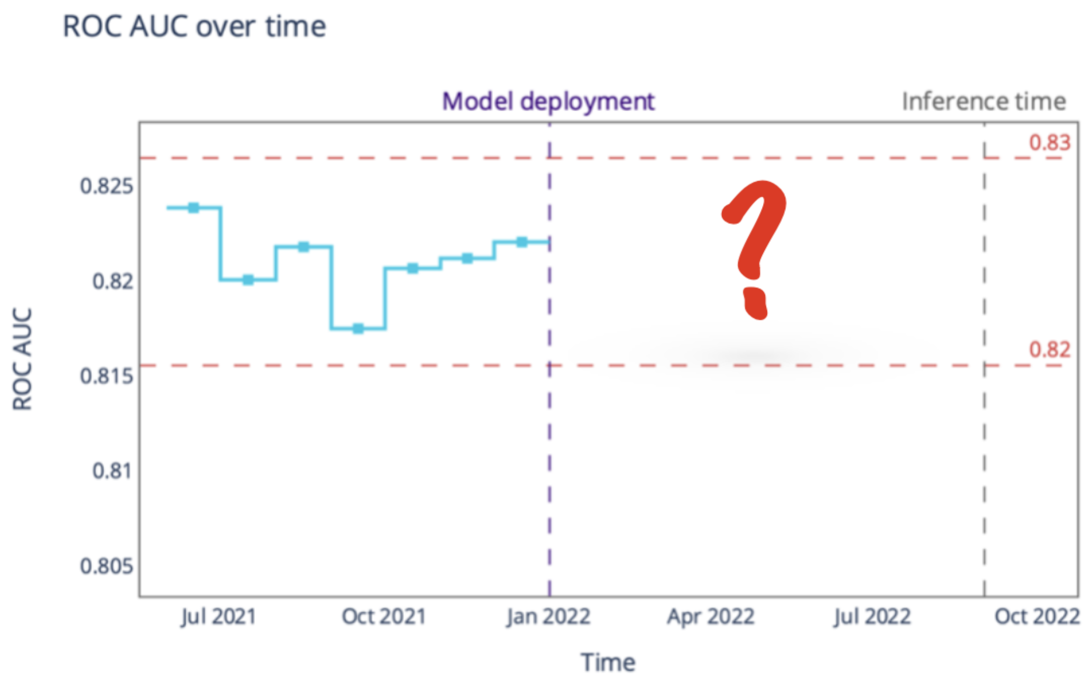
- Machine Learning Monitoring Concepts topics:
  - The ideal monitoring workflow
  - Challenges of monitoring ML models in production
  - Two silent model failures, covariate shift, and concept drift
  - Six methods for detecting covariate shift
  - The theoretical concepts behind CBPE and DLE performance estimation methods

What this course will cover?

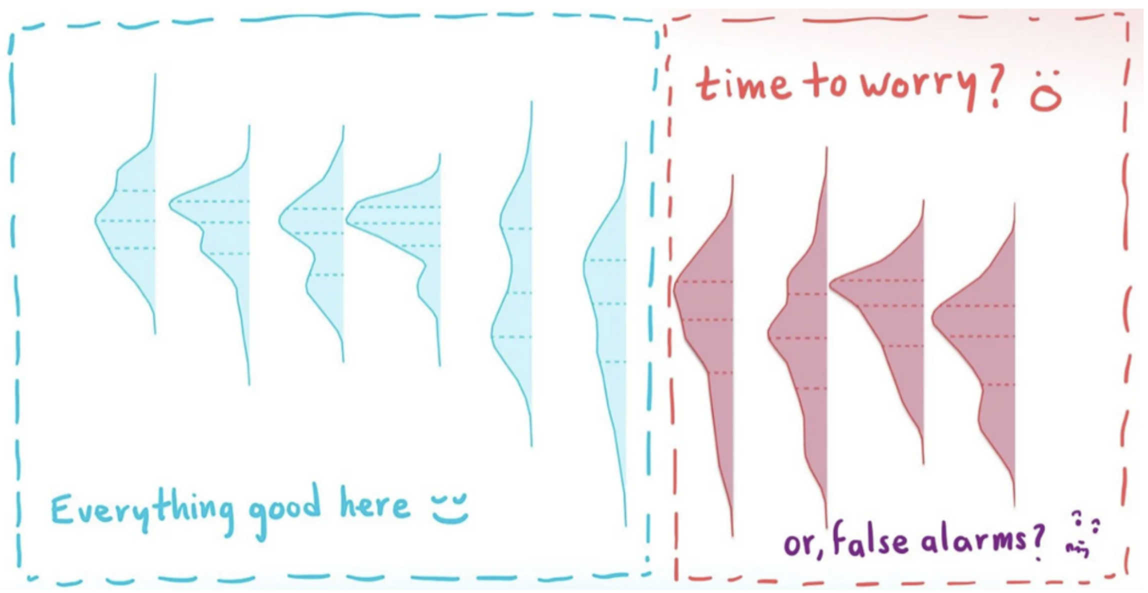
- Build a robust monitoring system using NannyML
- Implement performance estimation algorithms
- Use the business calculator to determine the monetary value of your machine learning model
- Run univariate and multivariate covariate shift detection methods

Monitoring challenges

No access to ground truth



Alerts fatigue



Open-source solution

nannyML

Trusted by data scientists at

Walmart yellow.ai TUI Allianz Razorpay

UBS DeepMind Risika euroclear KreditBee

Works with any ML framework

scikit learn LightGBM dmlc XGBoost

TensorFlow Keras PyTorch

The key features

1. Monitor what matter
   - Performance estimation and calculation
   - Business value estimation and calculation
2. Find what is broken
   - Univariate
   - Multivariate
   - Data quality
3. Fix it
   - Retraining triggers

How to use NannyML?


```python
import nannyml


# Load the dataset
reference, analysis, analysis_gt = nannyml.load_us_census_ma_employment_data()
```

Let's practice!

MONITORING MACHINE LEARNING IN PYTHON

Data preparation for NannyML

MONITORING MACHINE LEARNING IN PYTHON

Hakim Elakhrass
Co-founder and CEO of NannyML

Loading the data

```
dataset_name = "green_taxi_dataset.csv"
data = pd.read_csv(dataset_name)
data.head()
```

	lpep_pickup_datetime	PULocationID	DOLocationID	trip_distance	VendorID	payment_type	fare_amount	tip_amount
0	2016-12-01 00:13:25	225	65	2.79	2	2	11.0	0.00
1	2016-12-01 00:06:47	255	255	0.45	2	1	3.5	0.96
2	2016-12-01 00:29:45	41	42	1.20	1	3	6.0	0.00
3	2016-12-01 00:05:43	80	255	1.40	1	2	6.5	0.00
4	2016-12-01 00:47:13	255	189	3.50	1	1	13.5	3.70

Processing the data

```
# Create data partition
data['partition'] = pd.cut(
    data['lpep_pickup_datetime'],
    bins= [pd.to_datetime('2016-12-01'),
           pd.to_datetime('2016-12-08'),
           pd.to_datetime('2016-12-16'),
           pd.to_datetime('2017-01-01')],
    right=False,
    labels= ['train', 'test', 'prod']
)
```

Splitting the data

```python
# Target column name
target = 'tip_amount'
# Features column name
features = ["PULocationID", "DOLocationID", "trip_distance", "VendorID", "pickup_time"]
```

```python
# Train set
X_train = data.loc[data['partition'] == 'train', features]
y_train = data.loc[data['partition'] == 'train', target]

# Test set (later reference set)
X_test = data.loc[data['partition'] == 'test', features]
y_test = data.loc[data['partition'] == 'test', target]

# Production set (later analysis set)
X_prod = data.loc[data['partition'] == 'prod', features]
y_prod = data.loc[data['partition'] == 'prod', target]
```

Building the model

- Train `LGBMRegressor` using `lightgbm` library
- Evaluate the model on a test set
- Deploy the model

```
# Training the model
model = LGBMRegressor(random_state=42)
model.fit(X_train, y_train)

# Making predictions
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Evaluating the model on train and test set
mae_train = MAE(y_train, y_pred_train)
mae_test = MAE(y_test, y_pred_test)

# Deploying the model to production
y_pred_prod = model.predict(X_prod)
```

Creating reference and analysis sets

Reference period

- Uses a test set
- Requires ground truth
- Set the baseline performance

Analysis period

- Latest production data
- Ground truth is optional
- NannyML analyzes the data drift and the performance

```
# Creating reference set
reference = X_test.copy() # Test set features
reference['y_pred'] = y_pred_test # Predictions
reference['tip_amount'] = y_test # Labels
reference = reference.join(
    data['lpep_pickup_datetime']) # Timestamp
```

```
# Creating analysis set
analysis = X_prod.copy() # Production features
analysis['y_pred'] = y_pred_prod # Predictions
analysis = analysis.join(
    data['lpep_pickup_datetime']) # Timestamp
```

Reference set example

- **Timestamp** - the time when observation occurred (optional)
- **Features** - features fed to our model
- **Model outputs**
  - Predictions - prediction score outputted by the model
  - Prediction class labels - thresholded probability scores
- **Target** - contains ground truth

Features						Predictions	Targets	Timestamp
PULocationID	DOLocationID	trip_distance	VendorID	fare_amount	pickup_time	y_pred	tip_amount	lpep_pickup_datetime
112	40	6.93	2	24.5	0	4.920921	5.16	2016-12-08 00:00:00
7	226	1.50	2	12.0	0	2.048210	0.00	2016-12-08 00:00:00
223	223	1.43	2	6.5	0	1.575490	1.56	2016-12-08 00:00:01
112	37	3.25	2	14.5	0	2.810236	2.00	2016-12-08 00:00:03
112	69	10.40	1	36.0	0	7.068890	2.00	2016-12-08 00:00:05

Let's practice!

MONITORING MACHINE LEARNING IN PYTHON

Performance estimation

MONITORING MACHINE LEARNING IN PYTHON

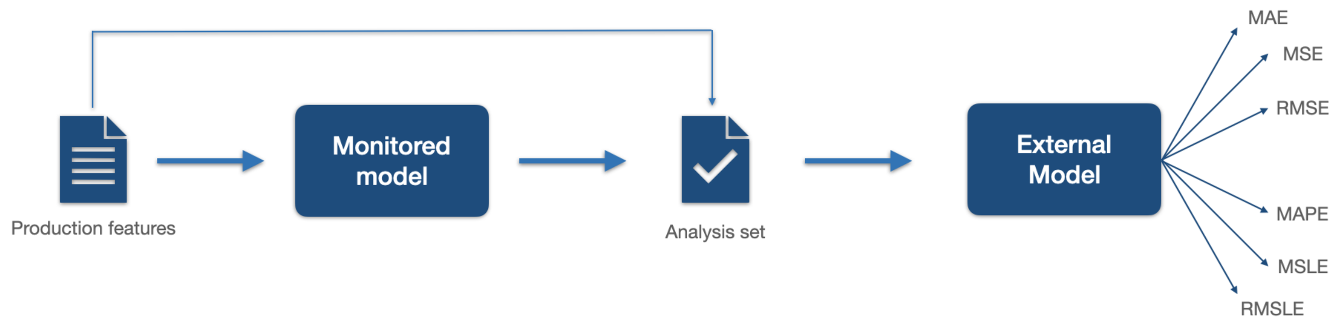
Hakim Elakhrass
Co-founder and CEO of NannyML

The algorithms

- **CBPE** - confidence based performance estimation
- **DLE** - direct loss estimation

Direct loss estimation

- Used for regression tasks
- Estimates loss function of monitored model
- LGBM is used as an "extra" model
- NannyML supports various regression metrics like MAE, MSE or RMSE



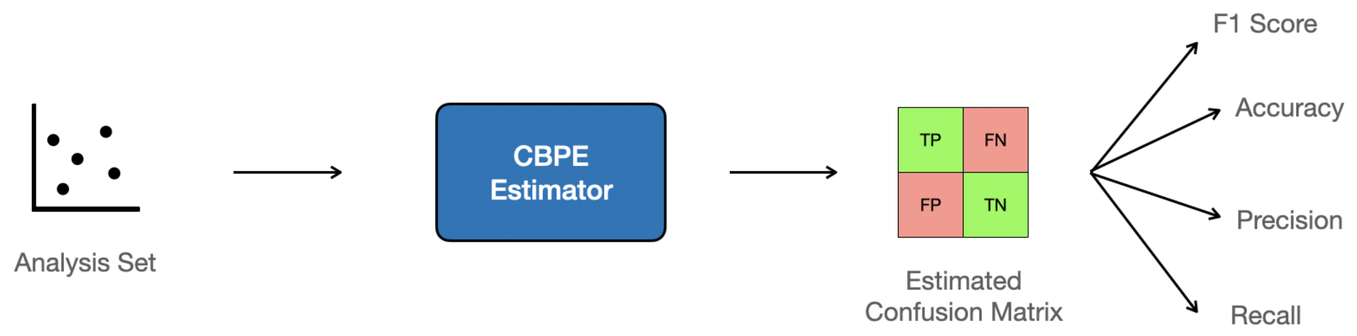
DLE - code implementation

```
# Initialize the DLE algorithm
estimator = nannyml.DLE(
    y_true='target',
    y_pred='y_pred',
    metrics=['rmse'],
    timestamp_column_name='timestamp',
    chunk_period='d'
    feature_column_names=features,
    tune_hyperparameters=False
)
```

```
# Fit the algorithm
estimator.fit(reference)
results = estimator.estimate(analysis)
```

Confidence based performance estimation

- Used for binary and multiclass classification problems
- Leverages confidence scores to estimate confusion matrix
- Estimates any classification performance metric



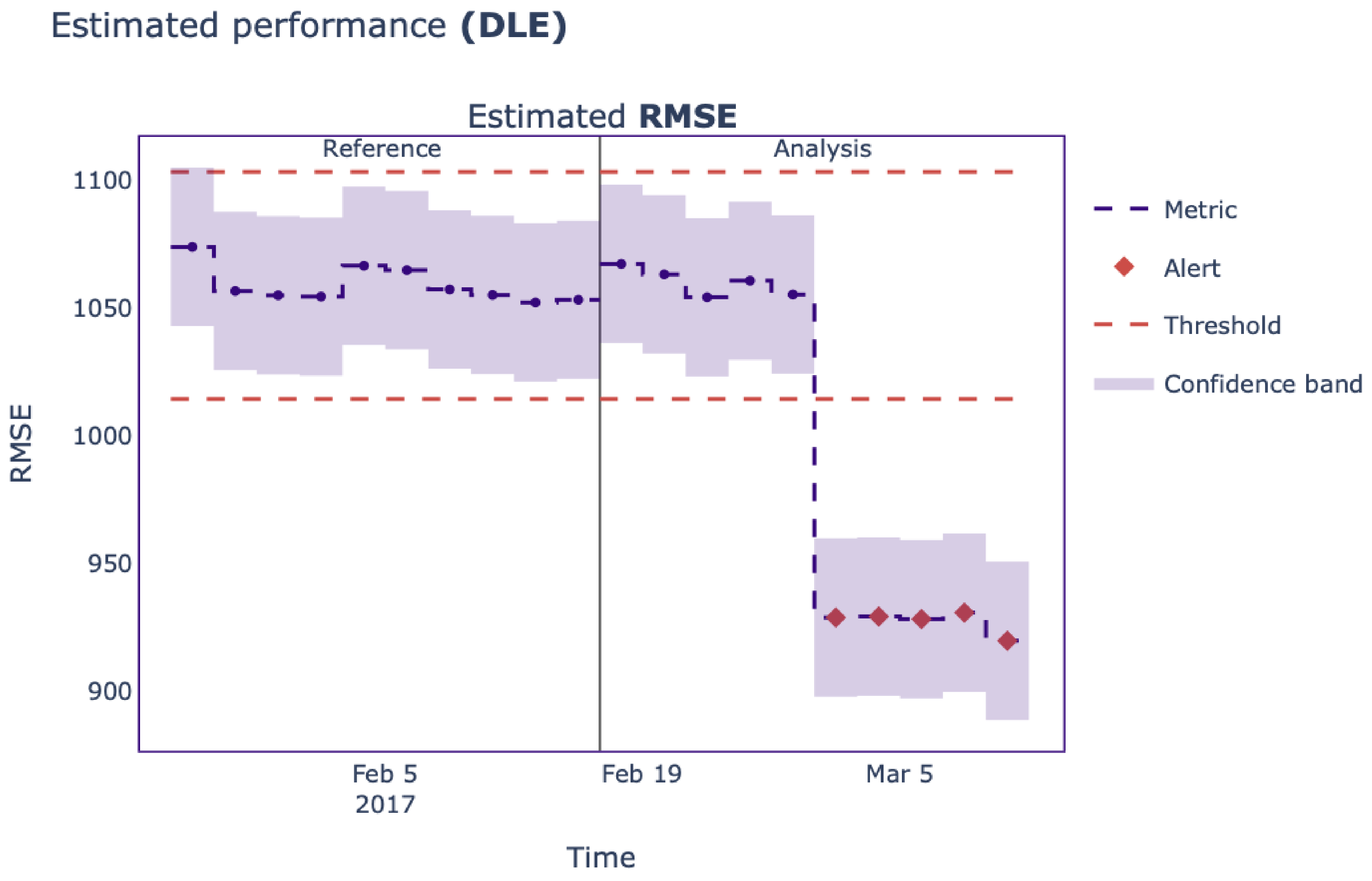
CBPE - code implementation

```python
# Initialize the CBPE algorithm
estimator = nannyml.CBPE(
    y_pred_proba='y_pred_proba',
    y_pred='y_pred',
    y_true='targets',
    timestamp_column_name='timestamp',
    metrics=['roc_auc'],
    chunk_period='d',
    problem_type='classification_binary',
)
```

```python
# Fit the algorithm
estimator.fit(reference)
results = estimator.estimate(analysis)
```

Results

```
results.plot().show()
```



Let's practice!

MONITORING MACHINE LEARNING IN PYTHON