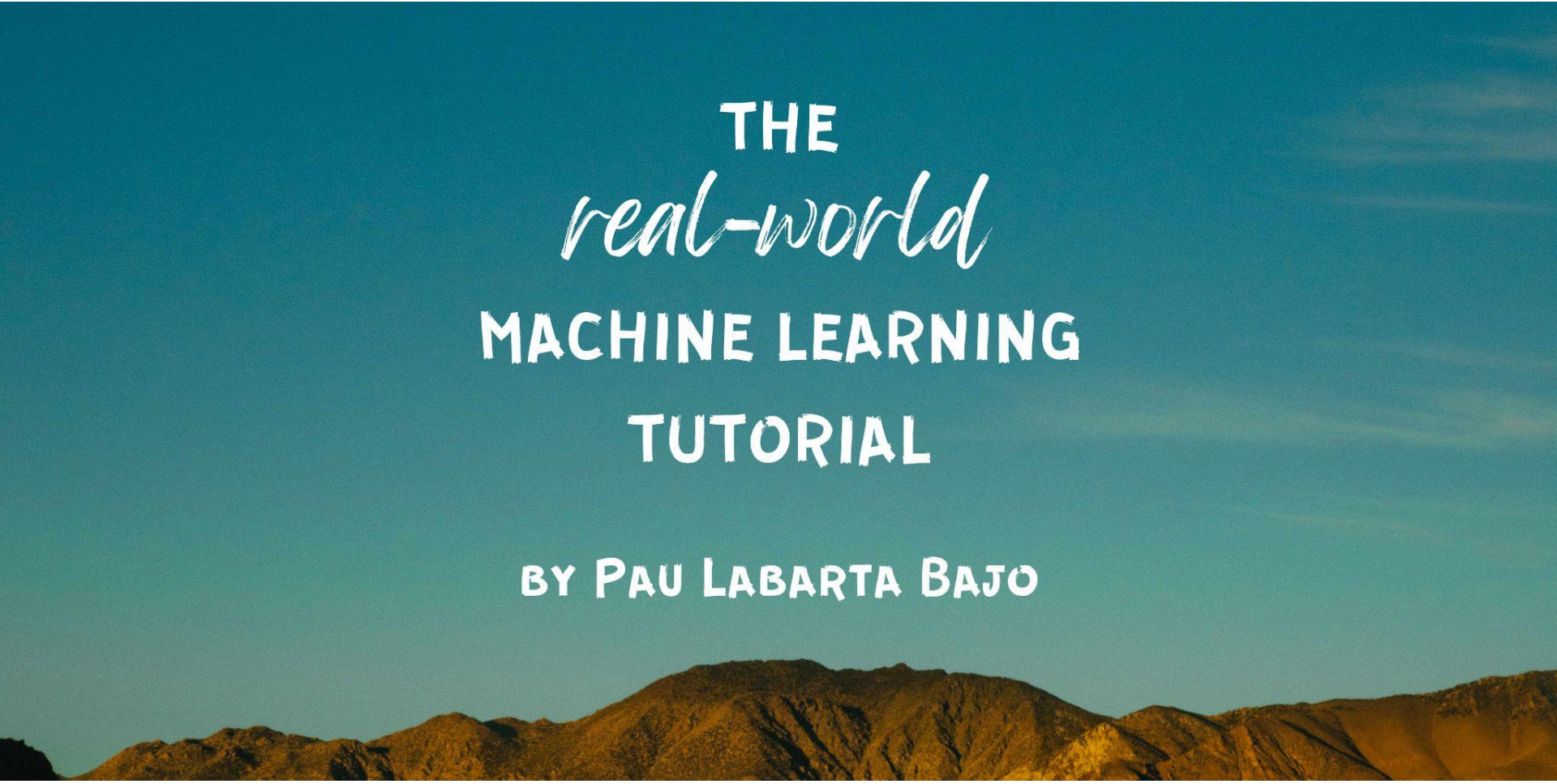

THE
real-world
MACHINE LEARNING
TUTORIAL
BY PAU LABARTA BAJO

Why is this course different?

In this course you will...

- Solve a **real-world** prediction problem.

In this course you will...

- Solve a **real-world** prediction problem.

- Build a **complete ML service**, as opposed to just training an ML model.

In this course you will...

- Solve a **real-world** prediction problem.

- Build a **complete ML service**, as opposed to just training an ML model.

- Learn all the tools you need to solve the problems **YOU CARE ABOUT**.

How are we going to get there?

Steps

---

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue

Steps

---

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue

2. **Data preparation** 📊
   From raw data to tabular data for Supervised ML

Steps

---

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue

2. **Data preparation** 📊
   From raw data to tabular data for Supervised ML

3. **Model training** 🏋️
   First without ML, then we add ML

Steps

---

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue

2. **Data preparation** 📊
   From raw data to tabular data for Supervised ML

3. **Model training** 🏋️
   First without ML, then we add ML.

4. **Model operationalization (MLOps)** 💰
   Without this our model has 0 business value

Steps

---

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue

2. **Data preparation** 📊
   From raw data to tabular data for Supervised ML

3. **Model training** 🏋️
   First without ML, then we add ML.

4. **Model operationalization (MLOps)** 💰
   Without this our model has 0 business value

Steps

1. **Business problem** 💼
   A ride-sharing company wants to maximize revenue
2. **Data preparation** 📊
   From raw data to tabular data for Supervised ML
3. **Model training** 🏋️
   First without ML, then we add ML.
4. **Model operationalization (MLOps)** 💰
   Without this our model has 0 business value

1. Business problem 💼

Business Problem

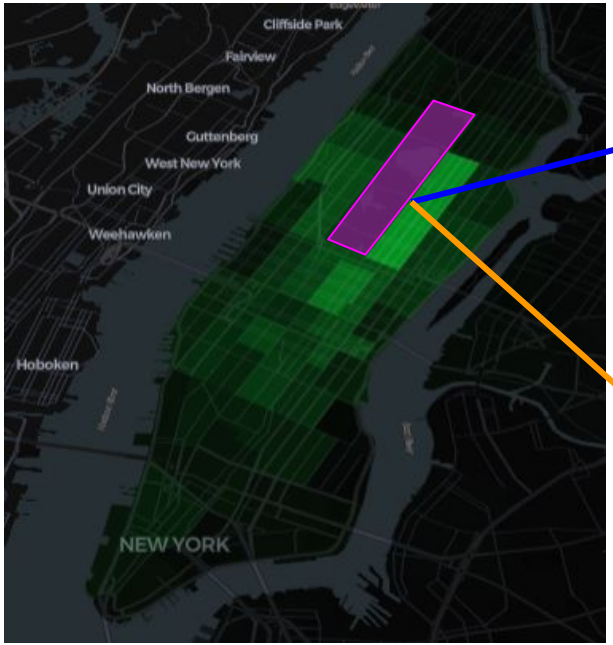
- You work as a data scientist 👩‍🔬🧑🏽‍🔬in a **ride-sharing app** company 🚗 (e.g. Uber)

Business Problem

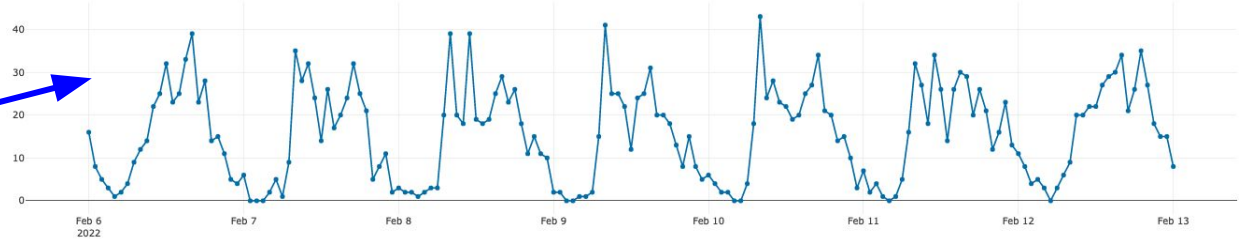
- You work as a data scientist 👩‍🔬🧑🏽‍🔬in a **ride-sharing app** company 🚗 (e.g. Uber)

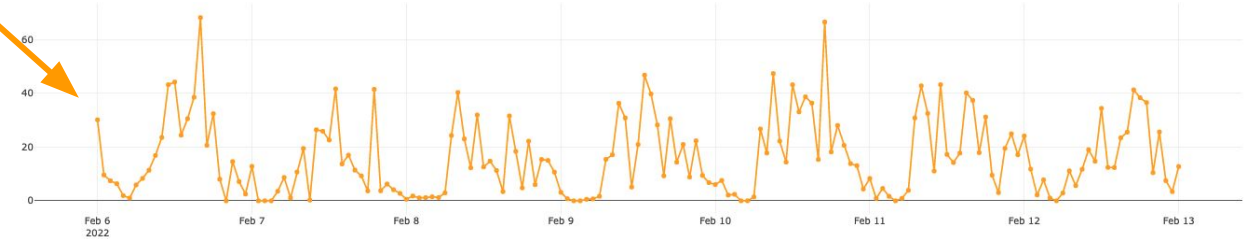
- Your job is to help the operations team **keep the fleet as busy as possible.**

Supply 🚗 and demand 🤦



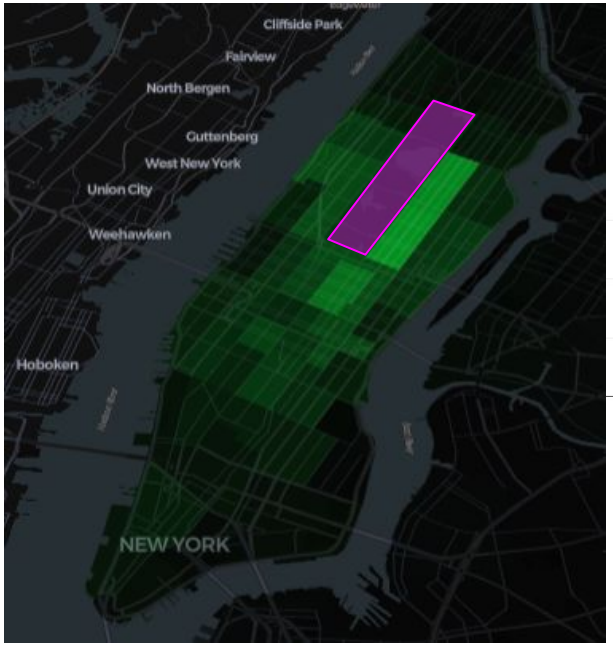
Drivers 🚗 around Central Park, hour by hour



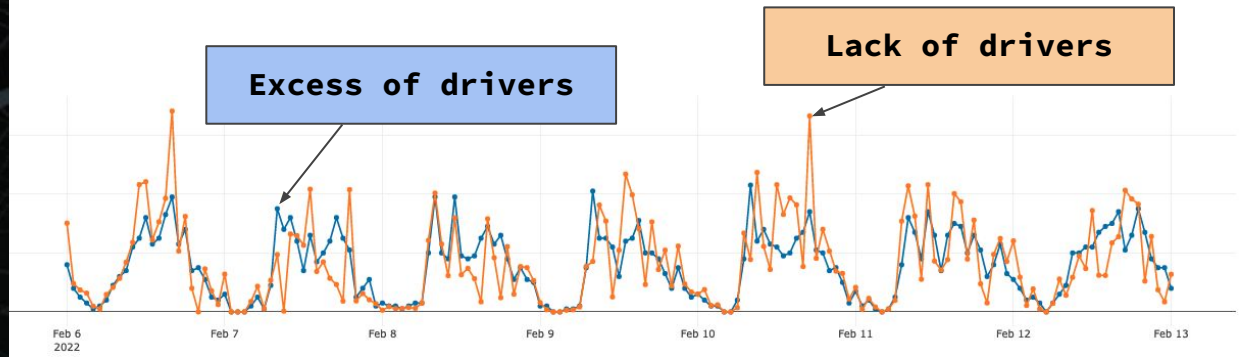


Users looking for a ride 🤦 around Central Park

Supply 🚗 and demand 🤦

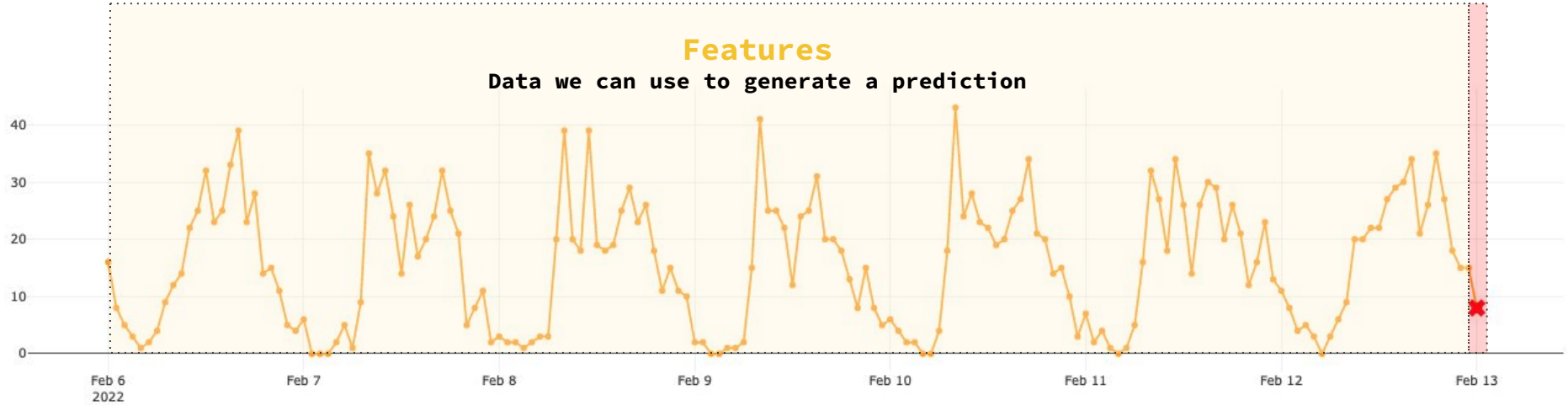


Drivers 🚗 around Central Park, hour by hour



Users looking for a ride 🤦 around Central Park

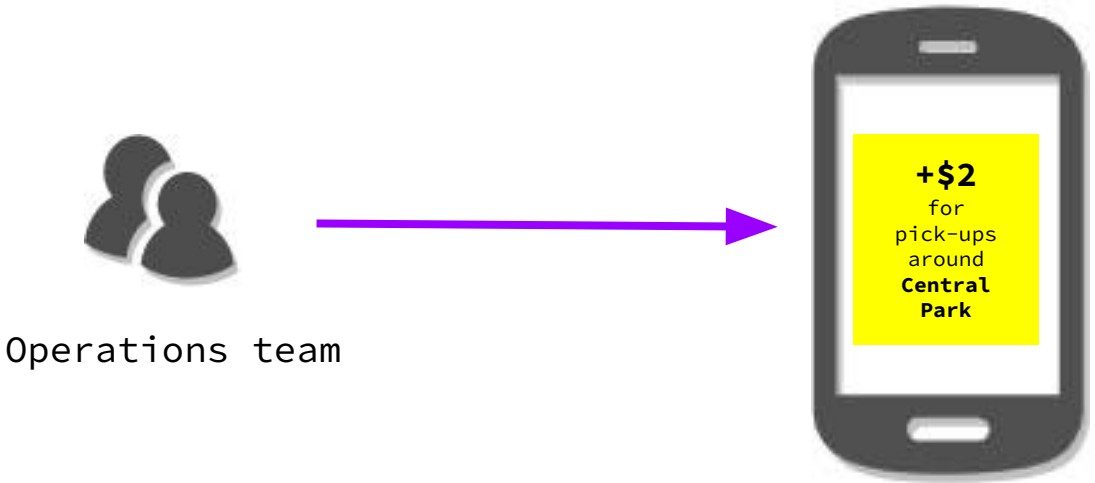
Let's predict user demand in the next hour



Target

This is what we want to predict
#users who will request a ride in the next hour

Ops team can adjust the distribution of the drivers

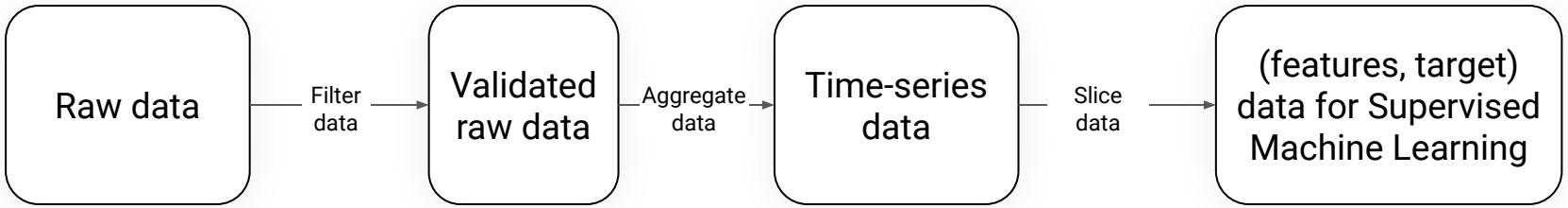


Development environment 🛠️

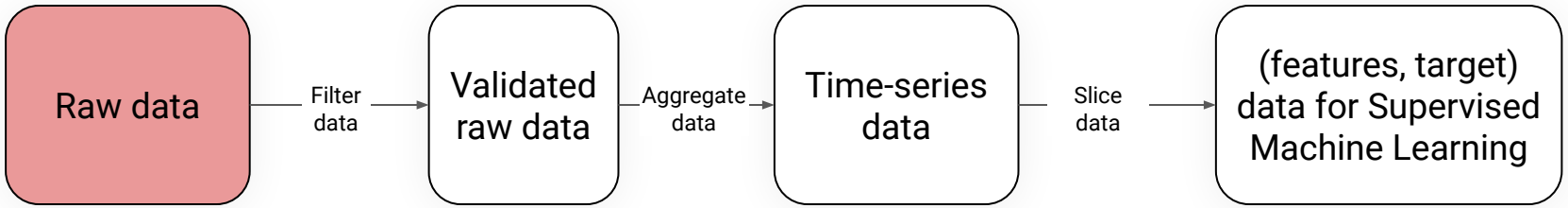
- **VSCode** for Python development
  - Other options: JupyterLab, PyCharm, Sublime…
- **Poetry** for virtual environments and packaging
  - Other options: virtualenv, conda, pipenv
- **Git** and GitHub for version control

2. Data preparation

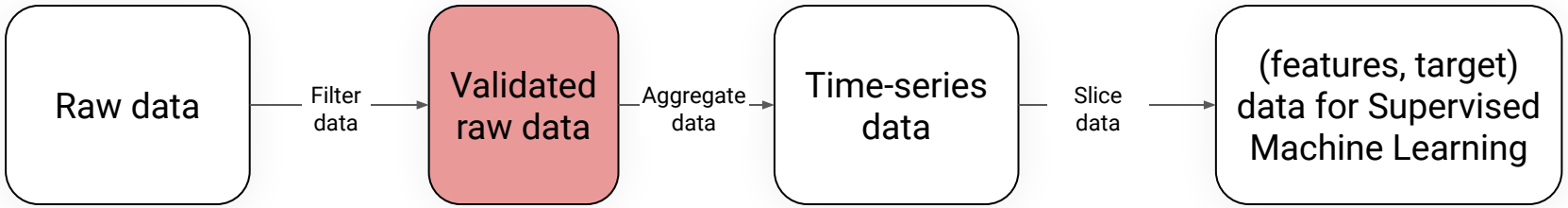
Data processing - From raw data to training data



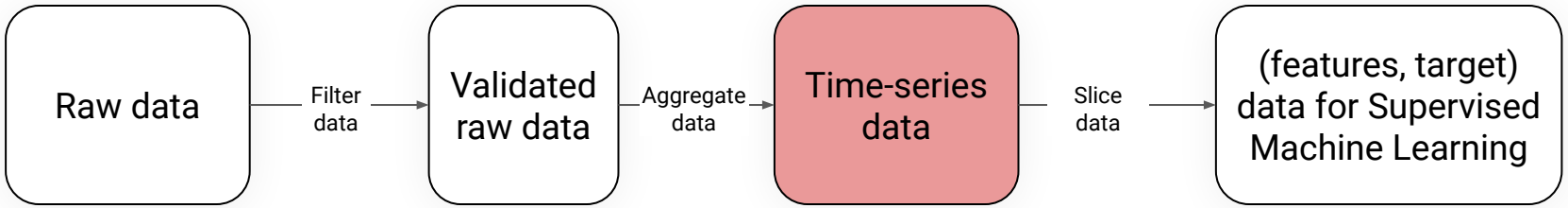
Data processing steps - from raw data to training data



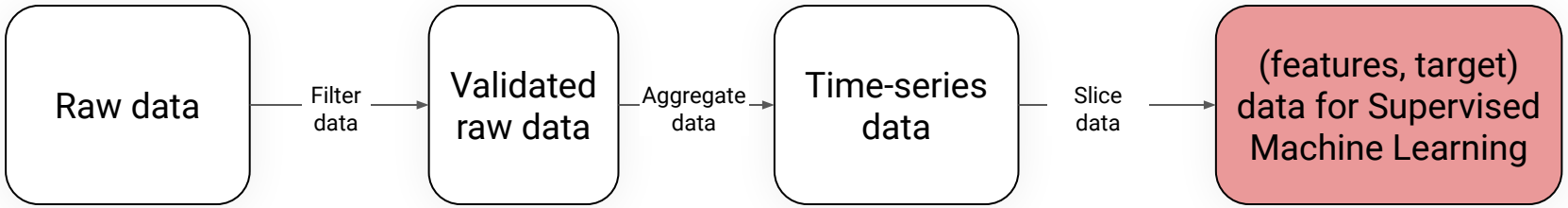
Data processing steps - from raw data to training data



Data processing steps - from raw data to training data

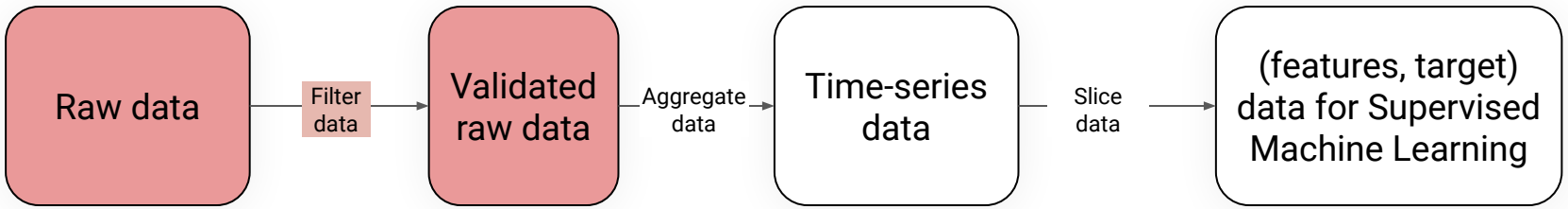


Data processing steps - from raw data to training data

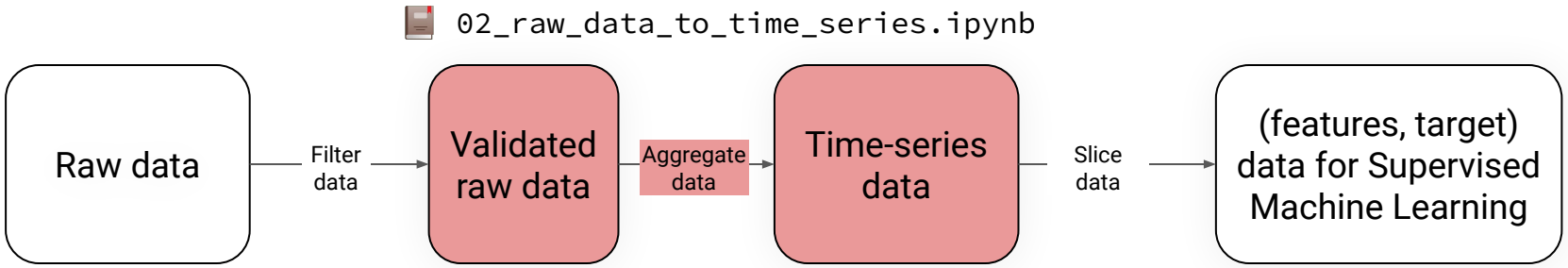


Step 1 - Data validation

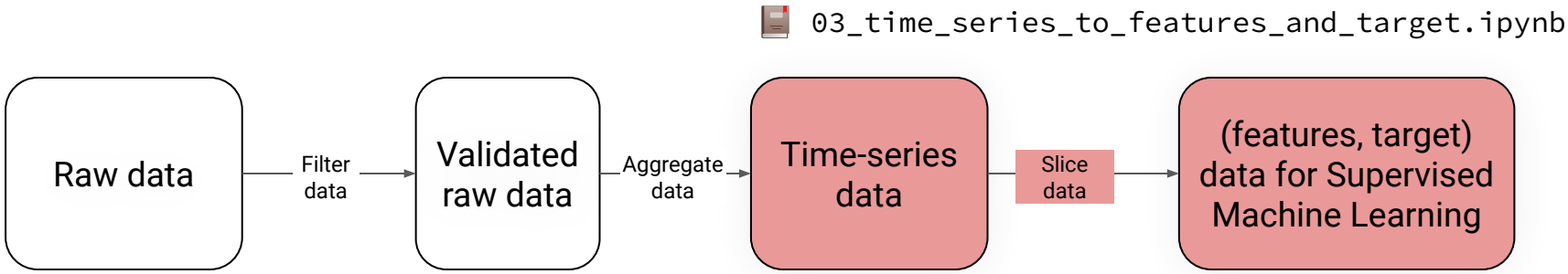
`01_load_and_clean_raw_data.ipynb`



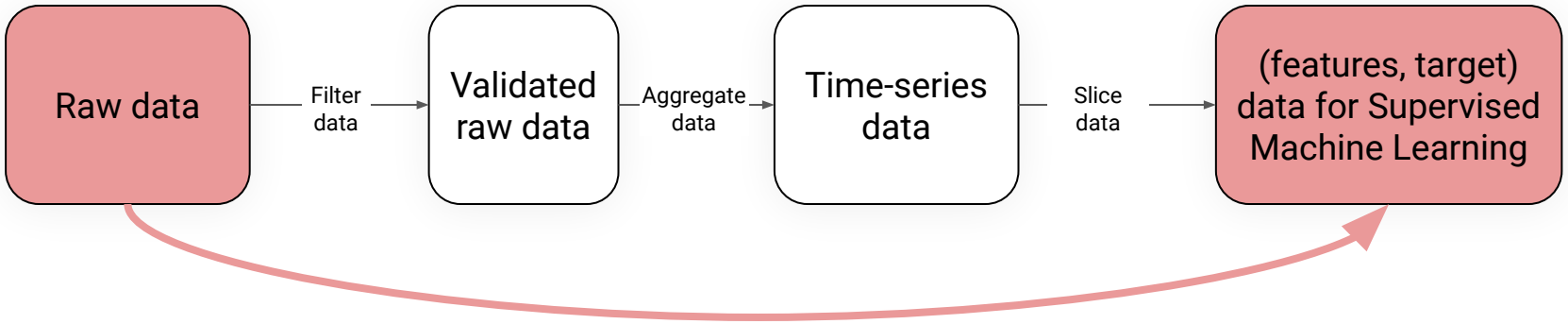
Step 2 - Raw data into time-series data



Step 3 - Time-series data into (features, target) data



Step 4 - From raw data to training data



📕 `04_raw_data_to_features_and_target.ipynb`

Time-series

location_id = 43

Slice & Slide

Supervised ML

Tabular data

	pickup_hour	rides
0	2022-06-01 00:00:00	3
1	2022-06-01 01:00:00	3
2	2022-06-01 02:00:00	4
3	2022-06-01 03:00:00	1
4	2022-06-01 04:00:00	3
5	2022-06-01 05:00:00	1
6	2022-06-01 06:00:00	13
7	2022-06-01 07:00:00	38
8	2022-06-01 08:00:00	79
9	2022-06-01 09:00:00	76
10	2022-06-01 10:00:00	105
11	2022-06-01 11:00:00	104
12	2022-06-01 12:00:00	111
13	2022-06-01 13:00:00	179
14	2022-06-01 14:00:00	177
15	2022-06-01 15:00:00	147
16	2022-06-01 16:00:00	154
17	2022-06-01 17:00:00	146
18	2022-06-01 18:00:00	145
19	2022-06-01 19:00:00	94
20	2022-06-01 20:00:00	138
21	2022-06-01 21:00:00	75
22	2022-06-01 22:00:00	61
23	2022-06-01 23:00:00	54
24	2022-06-02 00:00:00	27

n_features = 12

range(0, 12)

target

range(12, 13)

Cutoff indices

(0, 12, 13)

features

[3., 3., 4., ..., 76., 105., 104.]

target

111.

Time-series

location_id = 43

Slice & Slide

Supervised ML

Tabular data

	pickup_hour	rides
0	2022-06-01 00:00:00	3
1	2022-06-01 01:00:00	3
2	2022-06-01 02:00:00	4
3	2022-06-01 03:00:00	1
4	2022-06-01 04:00:00	3
5	2022-06-01 05:00:00	1
6	2022-06-01 06:00:00	13
7	2022-06-01 07:00:00	38
8	2022-06-01 08:00:00	79
9	2022-06-01 09:00:00	76
10	2022-06-01 10:00:00	105
11	2022-06-01 11:00:00	104
12	2022-06-01 12:00:00	111
13	2022-06-01 13:00:00	179
14	2022-06-01 14:00:00	177
15	2022-06-01 15:00:00	147
16	2022-06-01 16:00:00	154
17	2022-06-01 17:00:00	146
18	2022-06-01 18:00:00	145
19	2022-06-01 19:00:00	94
20	2022-06-01 20:00:00	138
21	2022-06-01 21:00:00	75
22	2022-06-01 22:00:00	61
23	2022-06-01 23:00:00	54
24	2022-06-02 00:00:00	27

n_features = 12

target

range(1, 13)

range(13, 14)

Cutoff indices

features

target

(0, 12, 13) [3., 3., 4., ..., 76., 105., 104.] 111.

(1, 13, 14) [3., 4., 1., ..., 105., 104., 111.] 179.

Time-series

location_id = 43

Slice & Slide

Supervised ML

Tabular data

	pickup_hour	rides
0	2022-06-01 00:00:00	3
1	2022-06-01 01:00:00	3
2	2022-06-01 02:00:00	4
3	2022-06-01 03:00:00	1
4	2022-06-01 04:00:00	3
5	2022-06-01 05:00:00	1
6	2022-06-01 06:00:00	13
7	2022-06-01 07:00:00	38
8	2022-06-01 08:00:00	79
9	2022-06-01 09:00:00	76
10	2022-06-01 10:00:00	105
11	2022-06-01 11:00:00	104
12	2022-06-01 12:00:00	111
13	2022-06-01 13:00:00	179
14	2022-06-01 14:00:00	177
15	2022-06-01 15:00:00	147
16	2022-06-01 16:00:00	154
17	2022-06-01 17:00:00	146
18	2022-06-01 18:00:00	145
19	2022-06-01 19:00:00	94
20	2022-06-01 20:00:00	138
21	2022-06-01 21:00:00	75
22	2022-06-01 22:00:00	61
23	2022-06-01 23:00:00	54
24	2022-06-02 00:00:00	27

n_features = 12

range(2, 14)

target

range(14, 15)

Cutoff indices	features	target
(0, 12, 13)	[3., 3., 4., ..., 76., 105., 104.]	111.
(1, 13, 14)	[3., 4., 1., ..., 105., 104., 111.]	179.
(2, 14, 15)	[4., 1., 3., ..., 104., 111., 179.]	177.

Time-series

location_id = 43

Slice & Slide

Supervised ML

Tabular data

	pickup_hour	rides
0	2022-06-01 00:00:00	3
1	2022-06-01 01:00:00	3
2	2022-06-01 02:00:00	4
3	2022-06-01 03:00:00	1
4	2022-06-01 04:00:00	3
5	2022-06-01 05:00:00	1
6	2022-06-01 06:00:00	13
7	2022-06-01 07:00:00	38
8	2022-06-01 08:00:00	79
9	2022-06-01 09:00:00	76
10	2022-06-01 10:00:00	105
11	2022-06-01 11:00:00	104
12	2022-06-01 12:00:00	111
13	2022-06-01 13:00:00	179
14	2022-06-01 14:00:00	177
15	2022-06-01 15:00:00	147
16	2022-06-01 16:00:00	154
17	2022-06-01 17:00:00	146
18	2022-06-01 18:00:00	145
19	2022-06-01 19:00:00	94
20	2022-06-01 20:00:00	138
21	2022-06-01 21:00:00	75
22	2022-06-01 22:00:00	61
23	2022-06-01 23:00:00	54
24	2022-06-02 00:00:00	27

n_features = 12

range(3, 15)

target

range(15, 16)

Cutoff indices

(0, 12, 13)

(1, 13, 14)

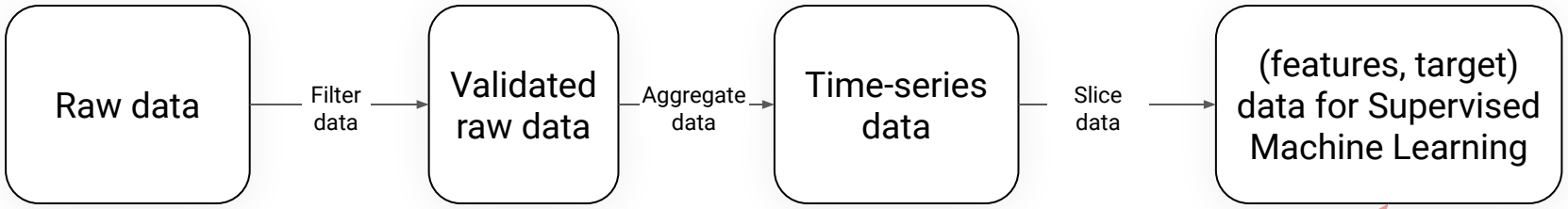
(2, 14, 15)

(3, 15, 16)

get_cutoff_indices()

features	target
[3., 3., 4., ..., 76., 105., 104.]	111.
[3., 4., 1., ..., 105., 104., 111.]	179.
[4., 1., 3., ..., 104., 111., 179.]	177.
[1., 3., 1., ..., 111., 179., 177.]	147.

Step 5 - Explore and visualize the final dataset

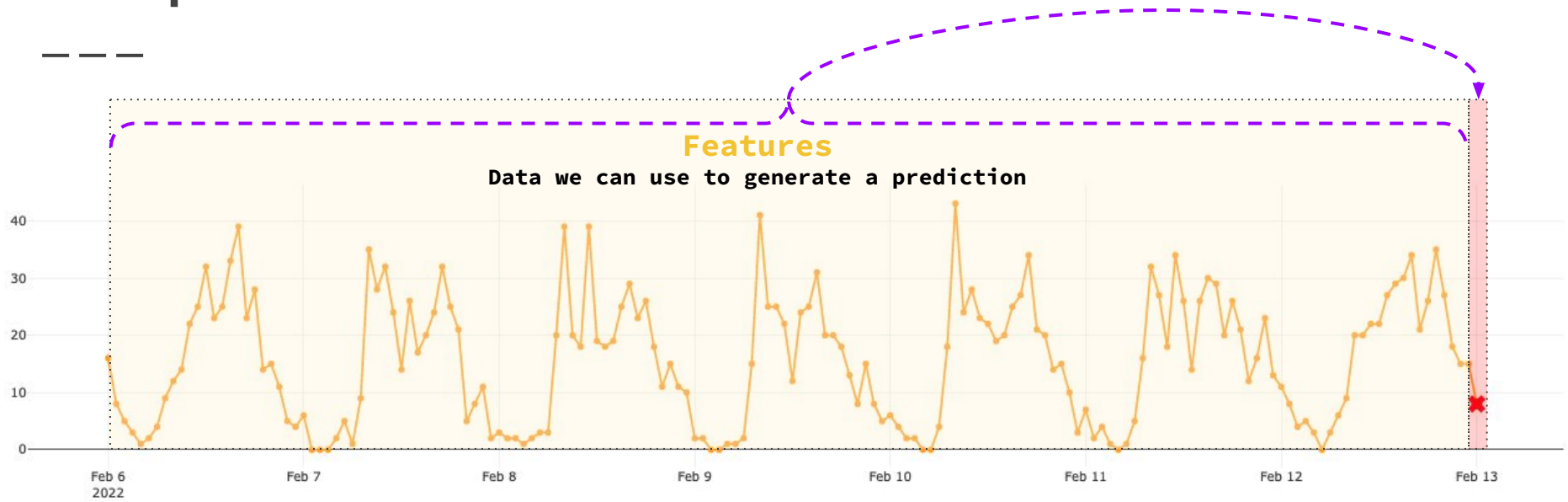


04_eda.ipynb

```
1 → Exploratory data analysis (EDA)
2 → Plot individual examples
```

3. Model training

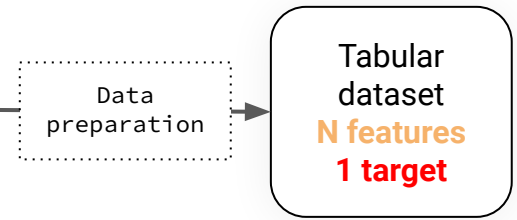
Let's predict user demand in the next hour



Target

This is what we want to predict
#users who will request a ride in the next hour

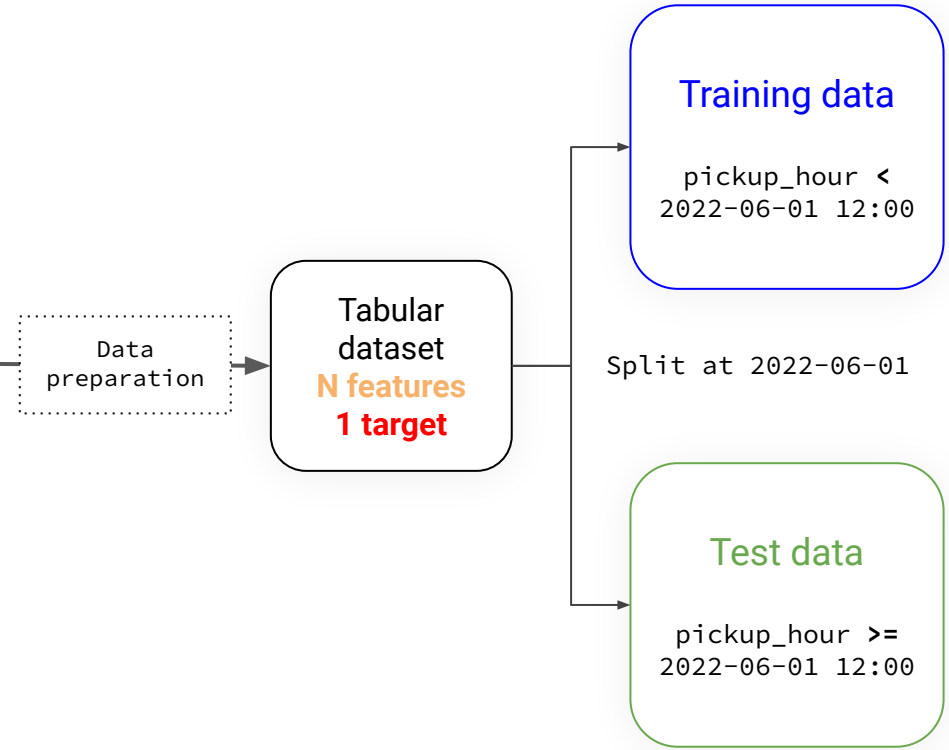
Split the data



features target

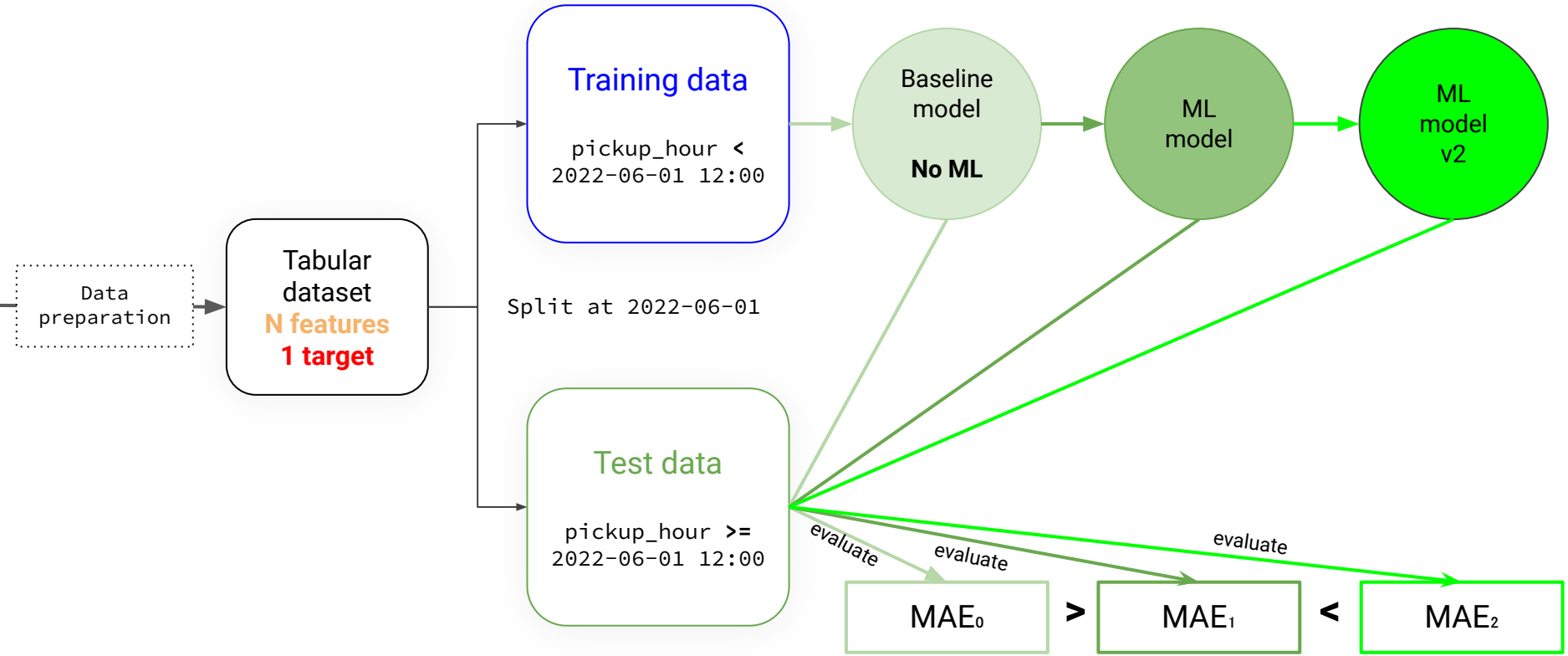
	rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago	pickup_hour	pickup_location_id	target_rides_next_hour
	341	452	...	232	2022-01-01 12:00	43	453
	...	...	...	...	...	...	...
	452	213	...	566	2022-06-01 13:00	211	322
	452	213	...	453	2022-06-01 12:00	43	212
	...	...	...	...	...	...	...
	29	34	...	54	2022-10-01 12:00	256	122

Split the data



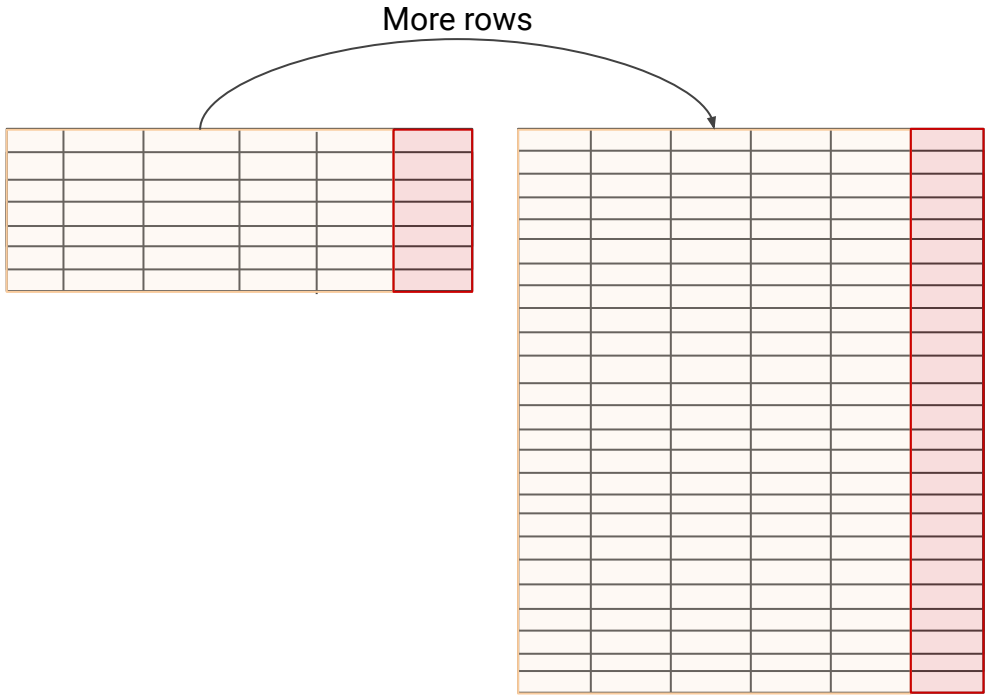
rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago	pickup_hour	pickup_location_id	target_rides_next_hour
341	452	...	232	2022-01-01 12:00	43	453
...	...	...	...	...	...	...
452	213	...	566	2022-06-01 13:00	211	322
452	213	...	453	**2022-06-01 12:00**	43	212
...	...	...	...	...	...	...
29	34	...	54	2022-10-01 12:00	256	122

How do you build a good ML model? Steps



4 ways to improve your model

1. Increase training data size

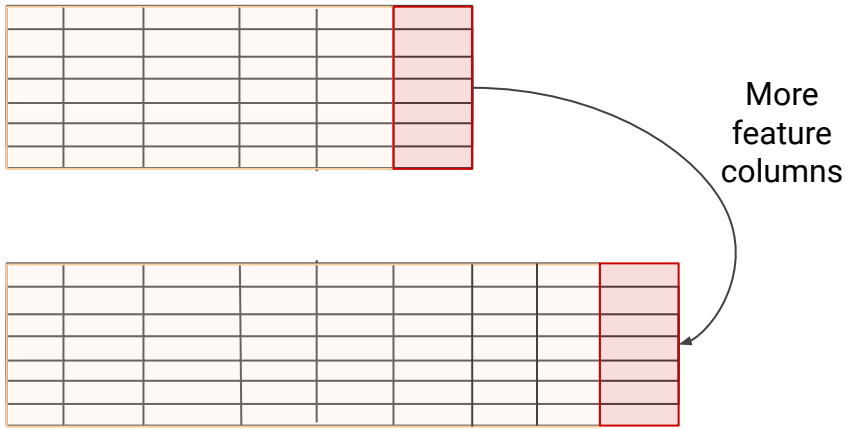


4 ways to improve your model

---

1. Increase training data size

2. **Add more features to the training data**



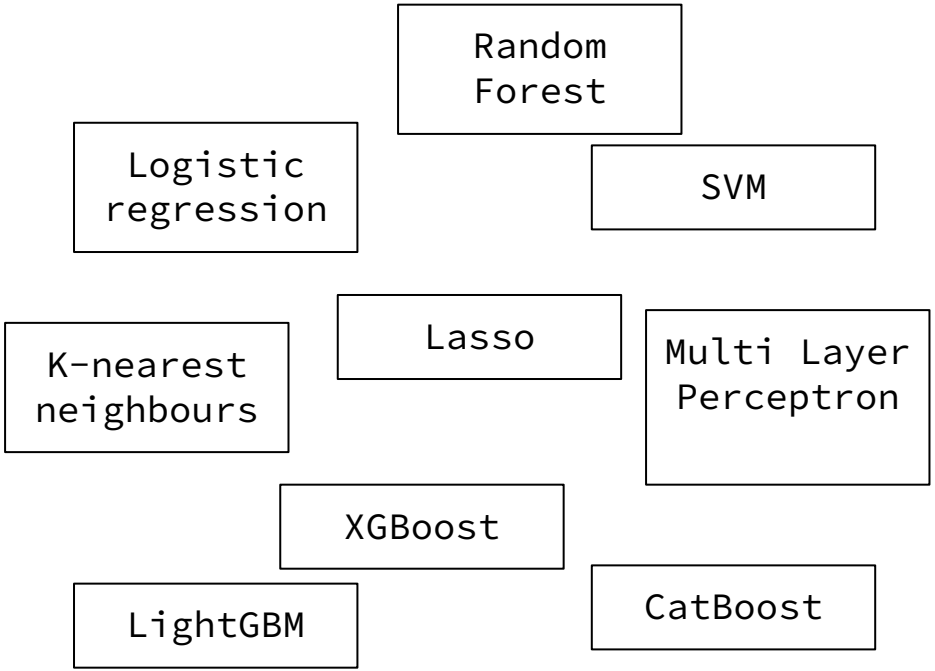
4 ways to improve your model

---

1. Increase training data size

2. Add more features to the training data

3. **Try another algorithm**



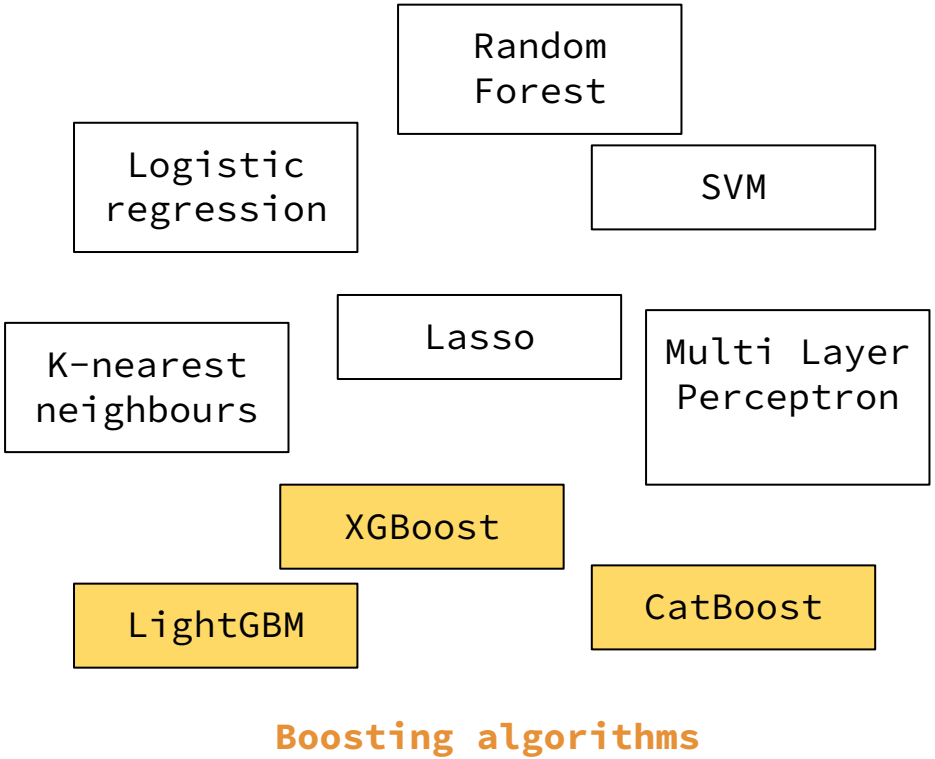
4 ways to improve your model

---

1. Increase training data size

2. Add more features to the training data

3. **Try another algorithm**



4 ways to improve your model

---

1. Increase training data size
2. Add more features to the training data
3. Try another algorithm
4. **Tune algorithm hyper-parameters**

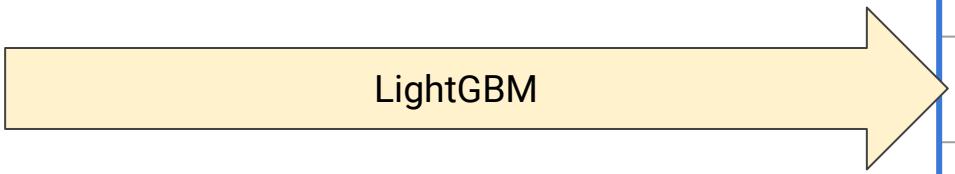
A few of XGBoost hyper-parameters

```
params = {
  'max_depth': trial.suggest_int('max_depth', 1, 9),
  'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0),
  'n_estimators': trial.suggest_int('n_estimators', 50, 500),
  'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
  'gamma': trial.suggest_loguniform('gamma', 1e-8, 1.0),
  'subsample': trial.suggest_loguniform('subsample', 0.01, 1.0),
  'colsample_bytree': trial.suggest_loguniform('colsample_bytree', 0.01, 1.0),
  'reg_alpha': trial.suggest_loguniform('reg_alpha', 1e-8, 1.0),
  'reg_lambda': trial.suggest_loguniform('reg_lambda', 1e-8, 1.0),
  'eval_metric': 'mlogloss',
  'use_label_encoder': False
}
```

Feature engineering

features

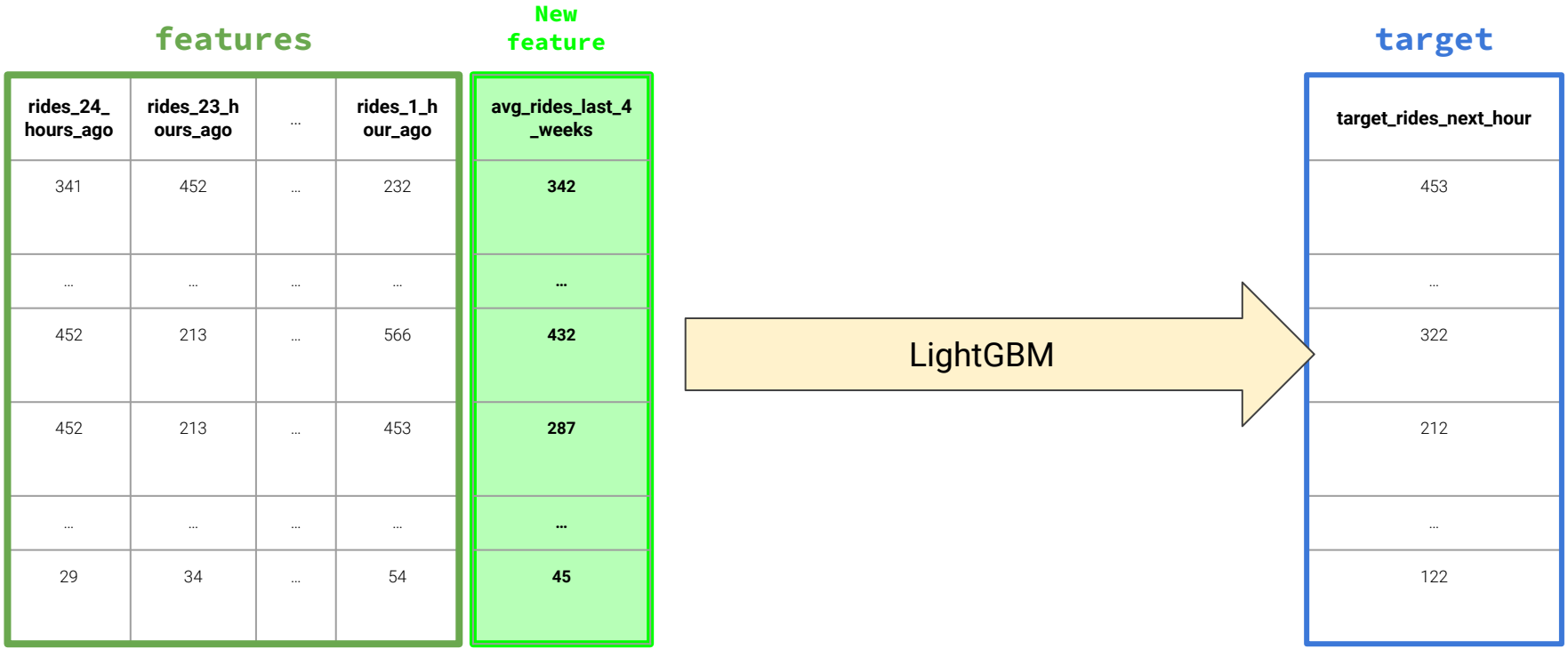
rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago
341	452	...	232
...	...	...	...
452	213	...	566
452	213	...	453
...	...	...	...
29	34	...	54



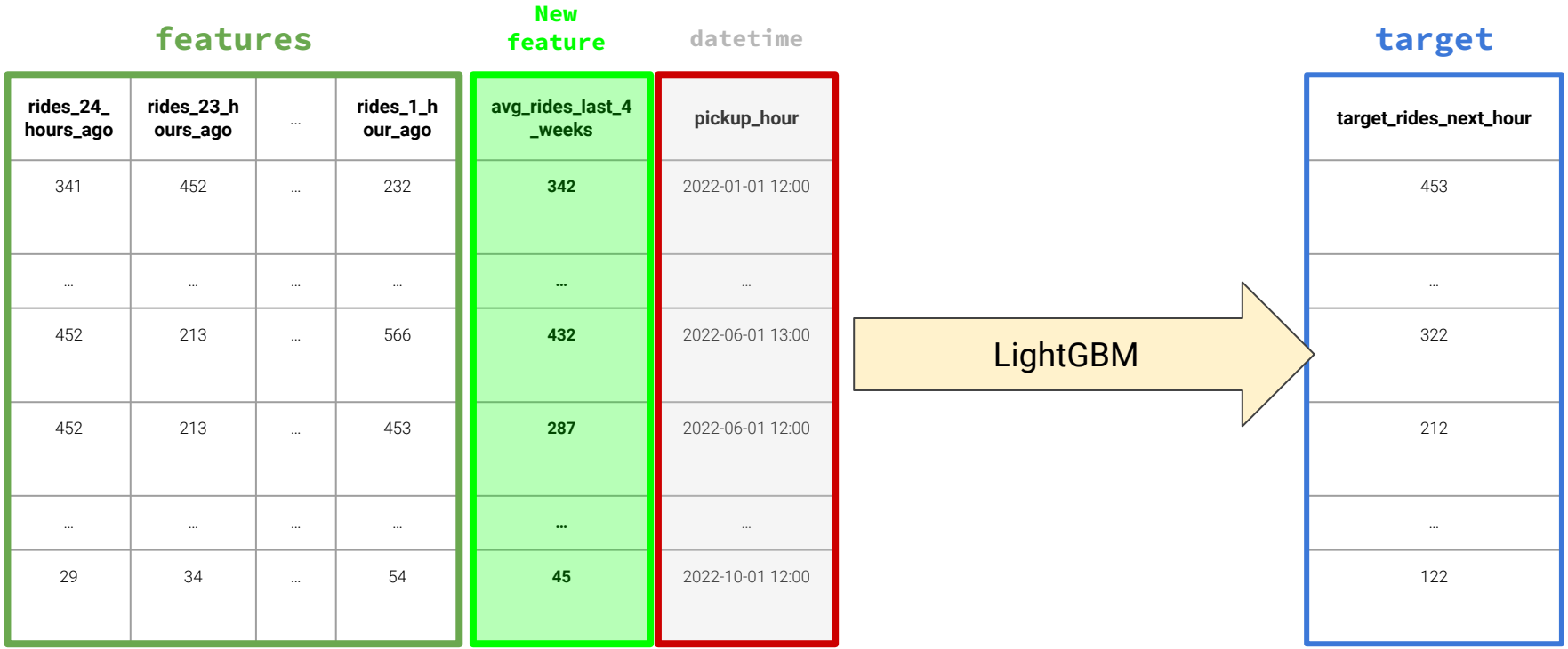
target

target_rides_next_hour
453
...
322
212
...
122

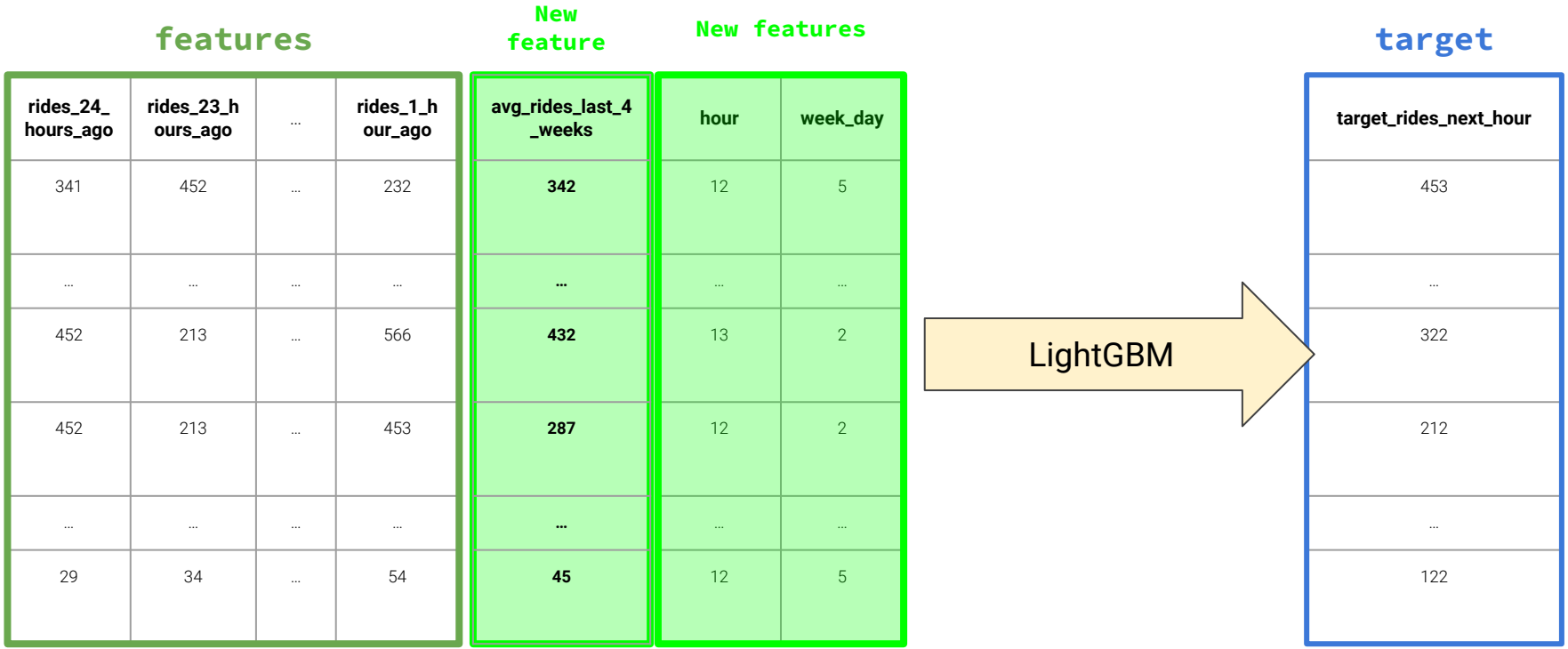
Idea #1 → Add average rides 7, 14, 21 and 28 days ago



Idea #2 → Extract cyclical features from `pick_hour`

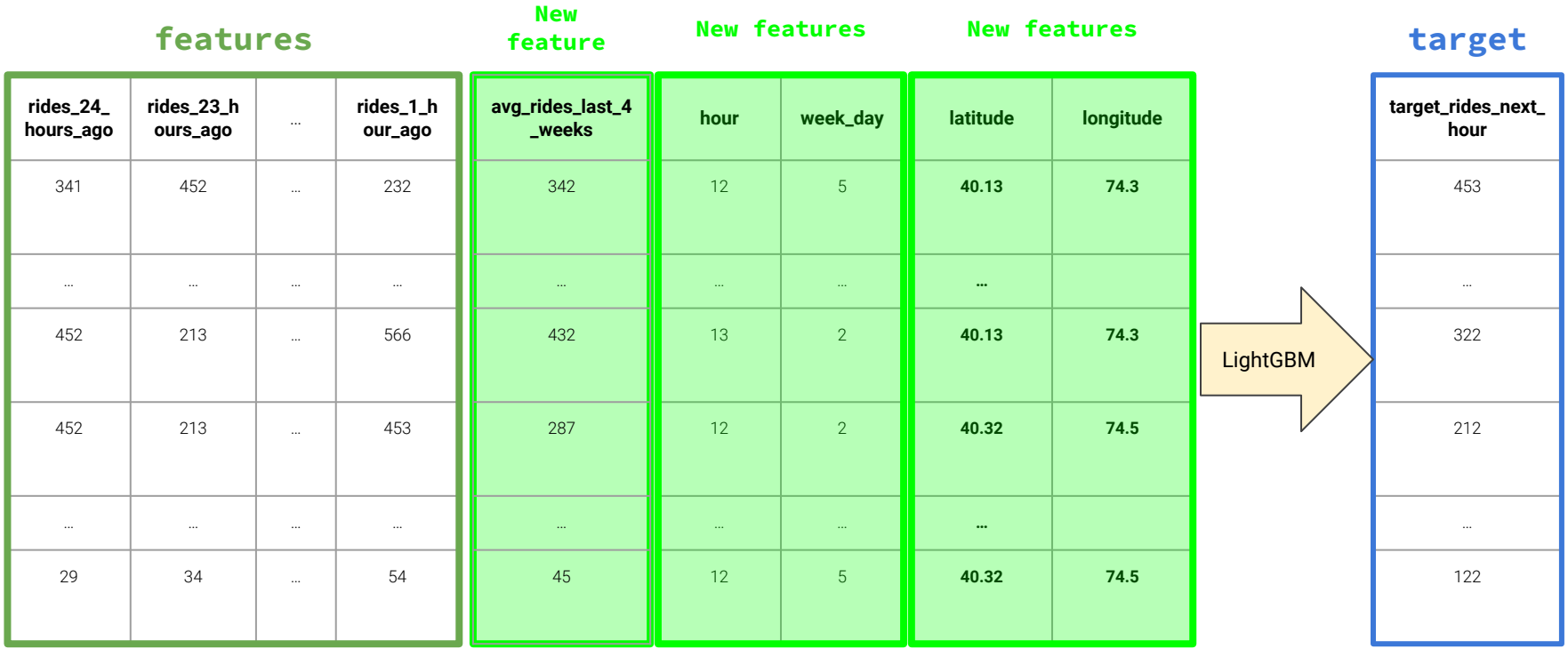


Idea #2 → Extract cyclical features from `pick_hour`

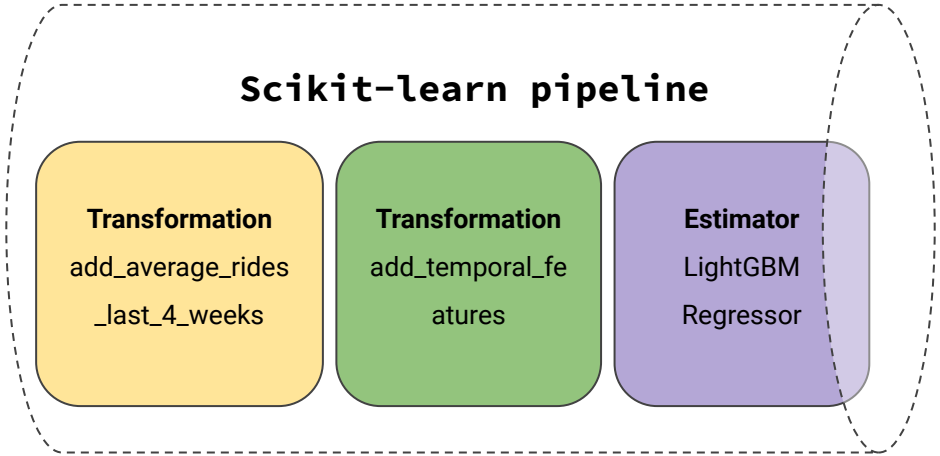


Idea #3 → Encode categorical feature `pickup_location_id`

Idea #3 → Encode categorical feature `pickup_location_id`

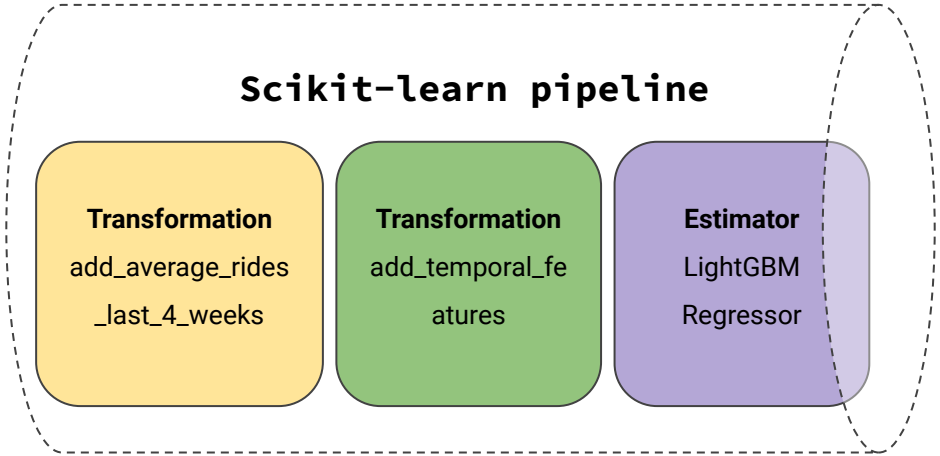


rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago
341	452	...	232
...	...	...	...
...	...	...	...
29	34	...	54



target_rides_next_hour
453
...
...
122

rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago
341	452	...	232
...	...	...	...
...	...	...	...
29	34	...	54



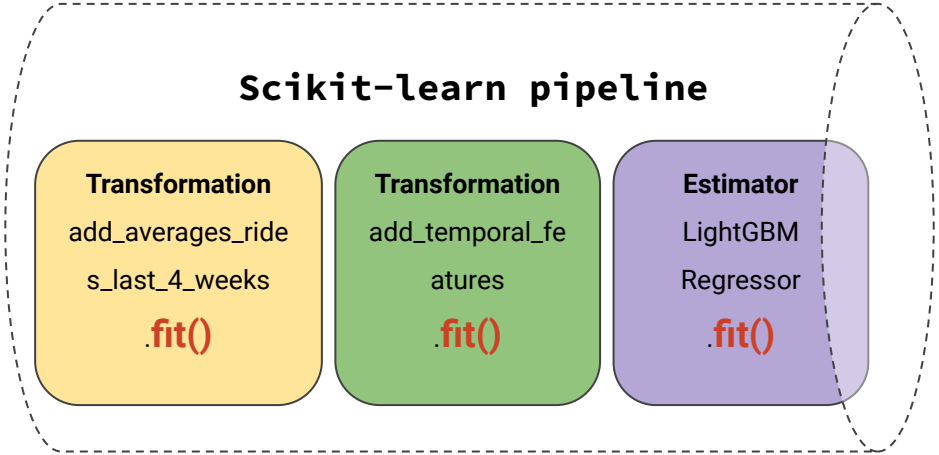
target_rides_next_hour
453
...
...
122

1. **Create** scikit-learn pipeline

```
from sklearn.pipeline import make_pipeline

pipeline = make_pipeline(
    add_average_rides_last_4_weeks,
    add_temporal_features(),
    LGBMRegressor()
)
```

X_train

rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago
341	452	...	232
...	...	...	...
...	...	...	...
29	34	...	54



y_train

target_rides_next_hour
453
...
...
122

1. **Create** scikit-learn pipeline

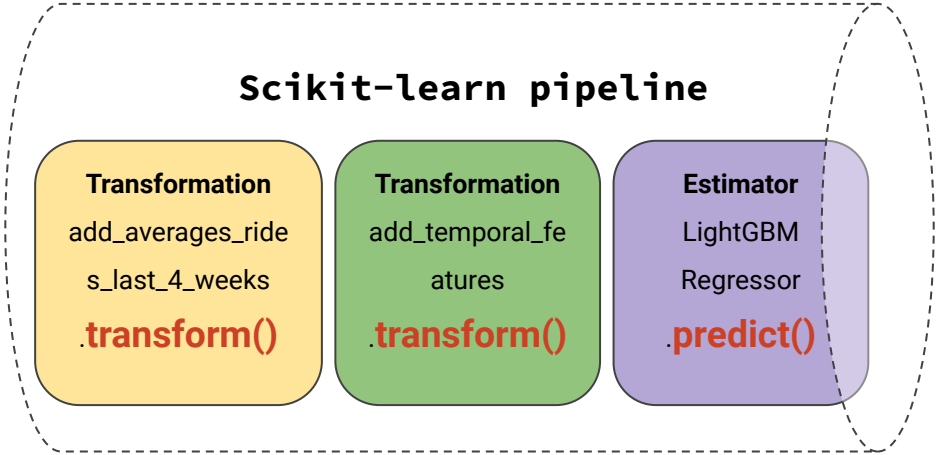
```
from sklearn.pipeline import make_pipeline

pipeline = make_pipeline(
    add_average_rides_last_4_weeks,
    add_temporal_features(),
    LGBMRegressor()
)
```

2. Train

```
pipeline.fit(X_train, y_train)
```

X_test

rides_24_hours_ago	rides_23_hours_ago	...	rides_1_hour_ago
341	452	...	232
...	...	...	...
...	...	...	...
29	34	...	54



predictions

target_rides_next_hour
453
...
...
122

1. **Create** scikit-learn pipeline

```
from sklearn.pipeline import make_pipeline

pipeline = make_pipeline(
    add_average_rides_last_4_weeks,
    add_temporal_features(),
    LGBMRegressor()
)
```

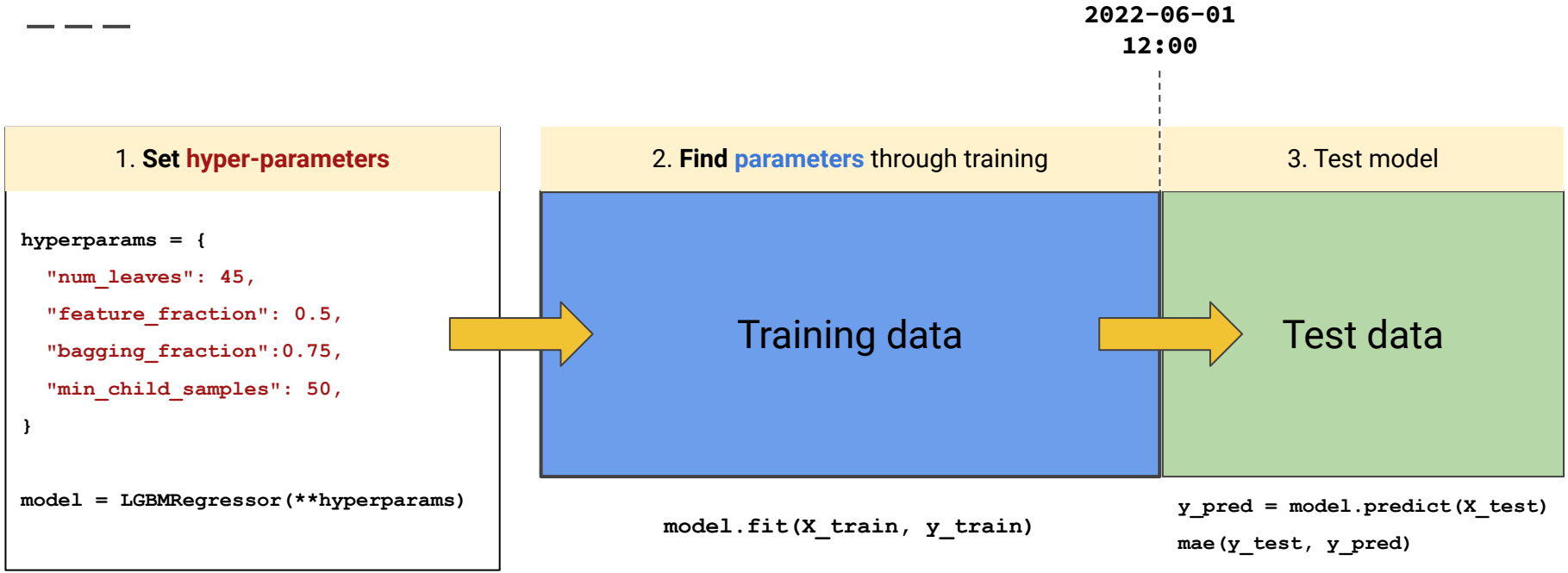
2. Train

```
pipeline.fit(X_train, y_train)
```

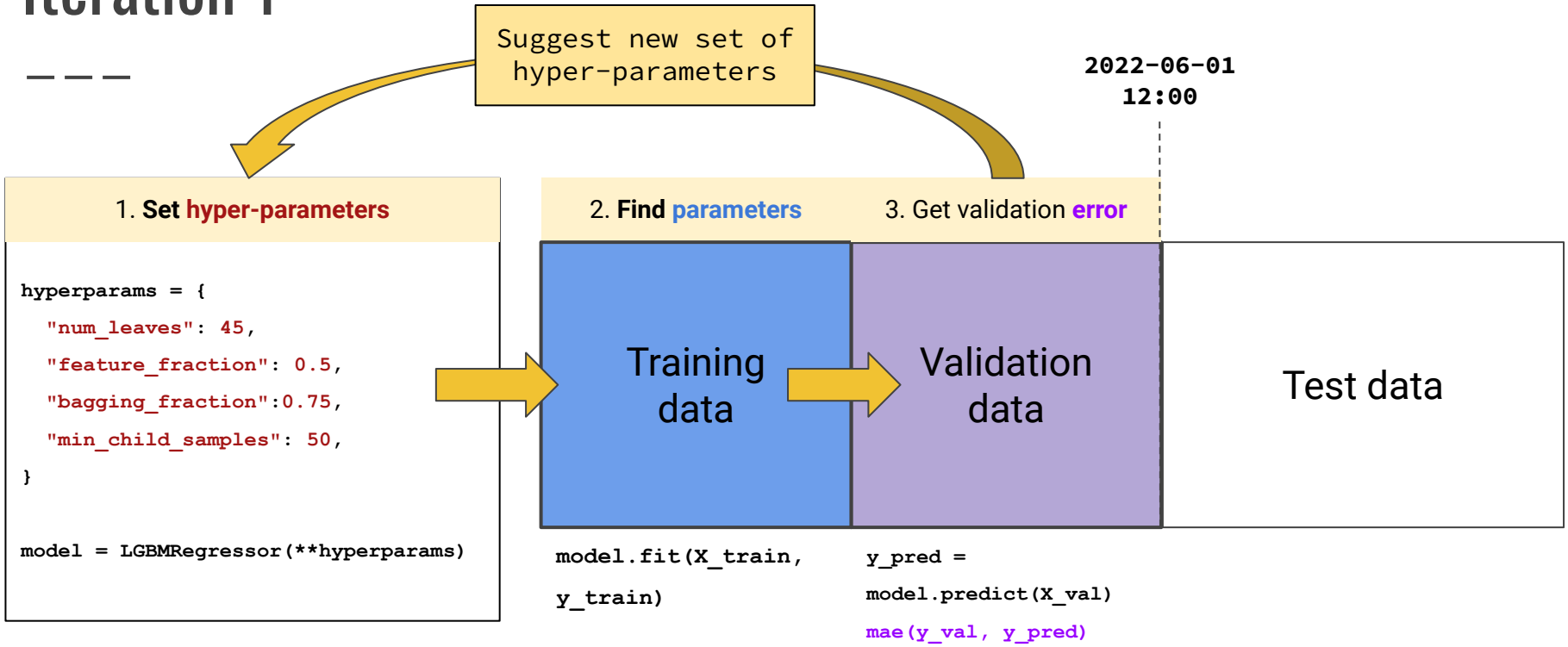
3. Predict

```
pipeline.predict(X_test)
```

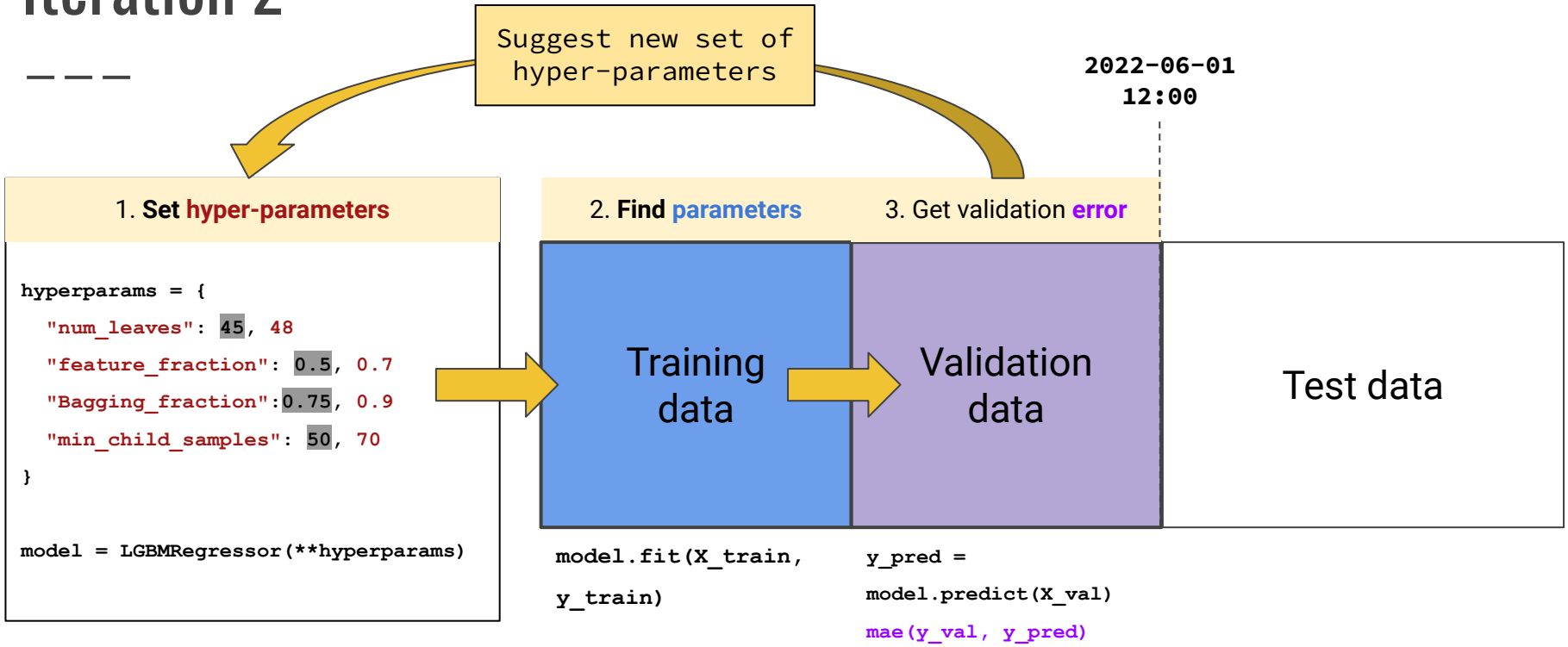
Hyper-parameter tuning



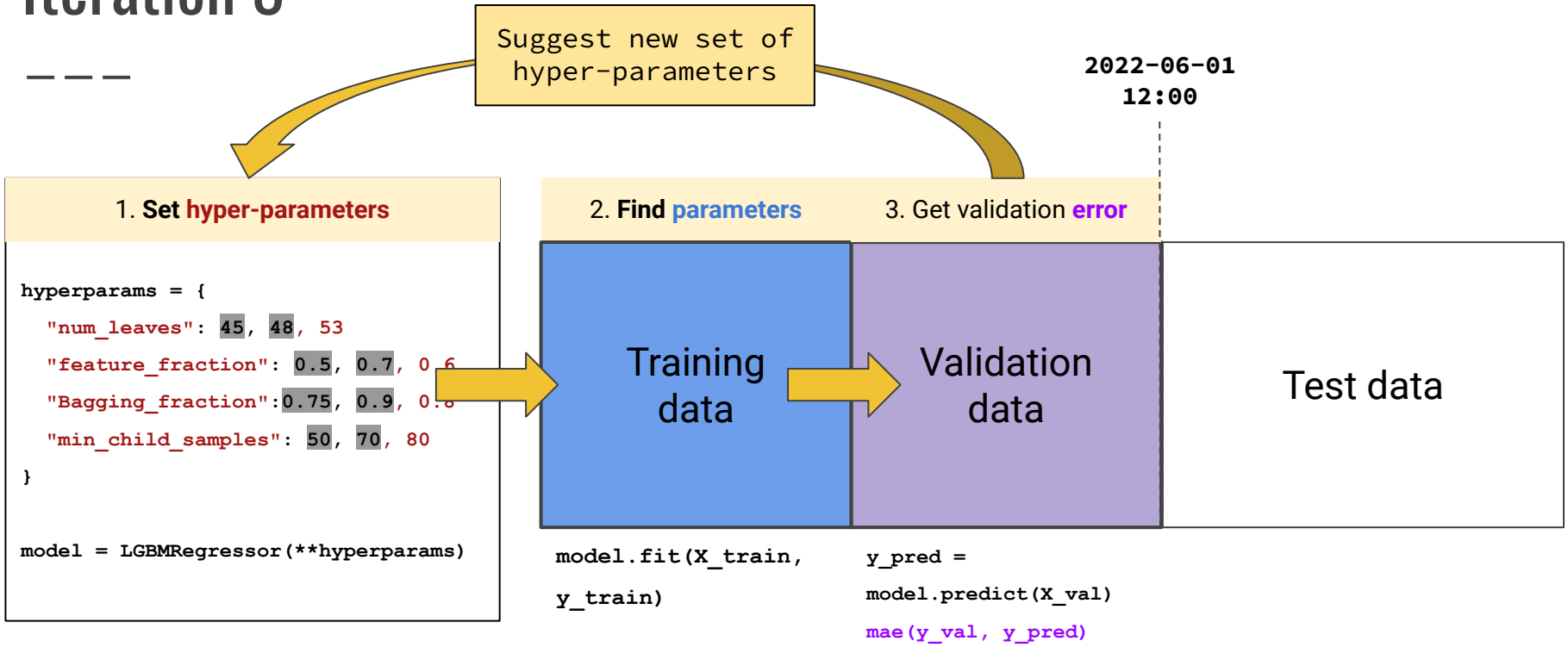
Iteration 1


Suggest new set of hyper-parameters
2022-06-01 12:00
1. Set hyper-parameters
2. Find parameters
3. Get validation error
hyperparams = {
"num_leaves": 45,
"feature_fraction": 0.5,
"bagging_fraction":0.75,
"min_child_samples": 50,
}
model = LGBMRegressor(**hyperparams)
Training data
Validation data
Test data
model.fit(X_train, y_train)
y_pred = model.predict(X_val)
mae(y_val, y_pred)

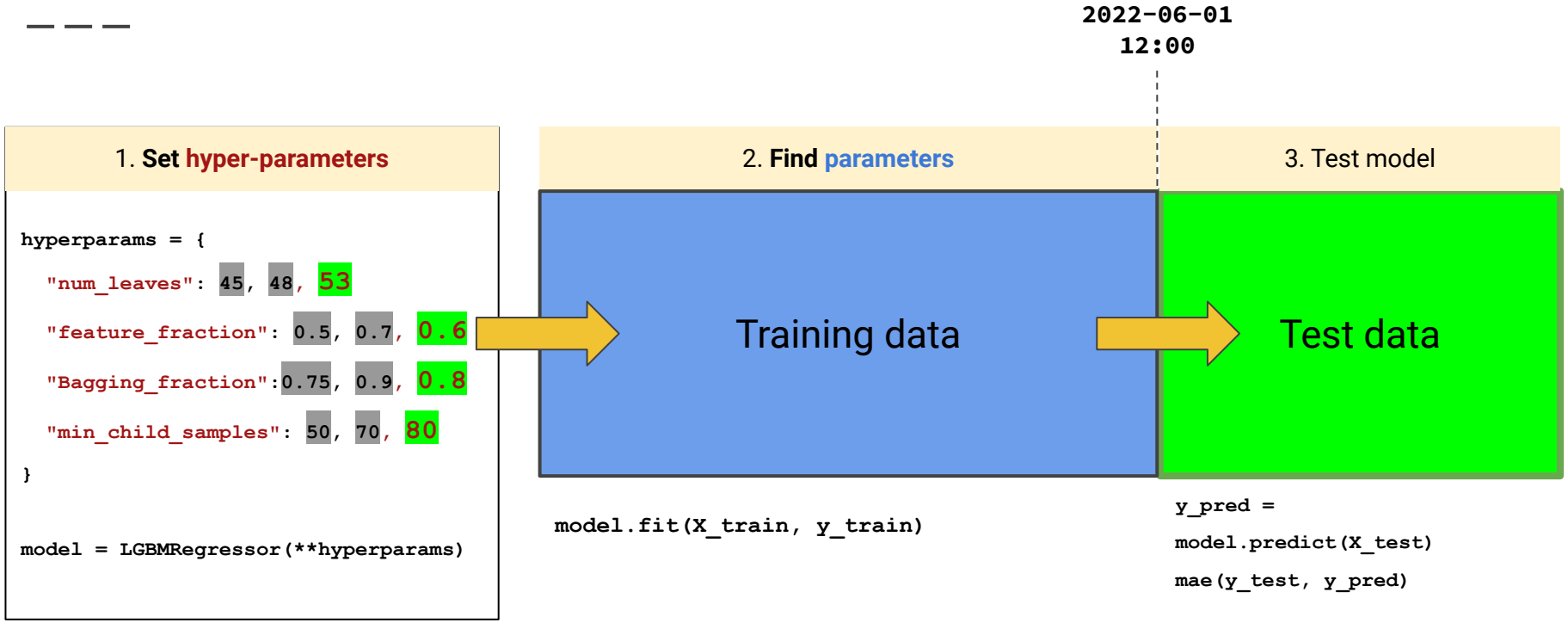
Iteration 2


Suggest new set of hyper-parameters
2022-06-01 12:00
1. Set hyper-parameters
2. Find parameters
3. Get validation error
hyperparams = {
"num_leaves": 45, 48
"feature_fraction": 0.5, 0.7
"Bagging_fraction":0.75, 0.9
"min_child_samples": 50, 70
}
model = LGBMRegressor(**hyperparams)
Training data
Validation data
Test data
model.fit(X_train, y_train)
y_pred = model.predict(X_val)
mae(y_val, y_pred)

Iteration 3


Suggest new set of hyper-parameters
2022-06-01 12:00
1. Set hyper-parameters
2. Find parameters
3. Get validation error
hyperparams = {
"num_leaves": 45, 48, 53
"feature_fraction": 0.5, 0.7, 0.6
"Bagging_fraction":0.75, 0.9, 0.8
"min_child_samples": 50, 70, 80
}
model = LGBMRegressor(**hyperparams)
Training data
Validation data
Test data
model.fit(X_train, y_train)
y_pred = model.predict(X_val)
mae(y_val, y_pred)

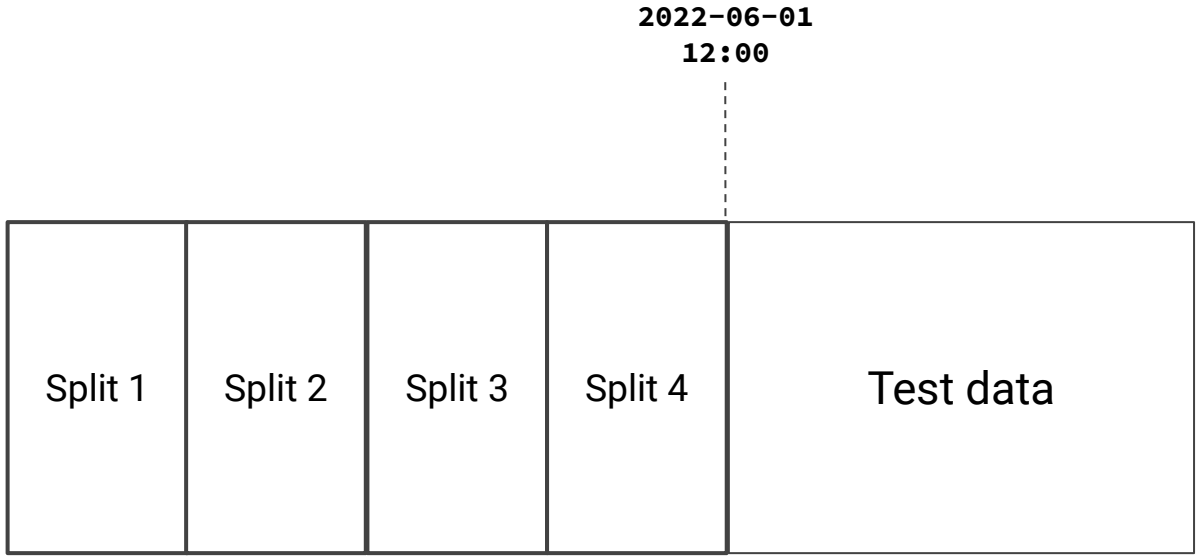
Use best hyper-parameters, train and test



N split for better results

1. **Set hyper-parameters**

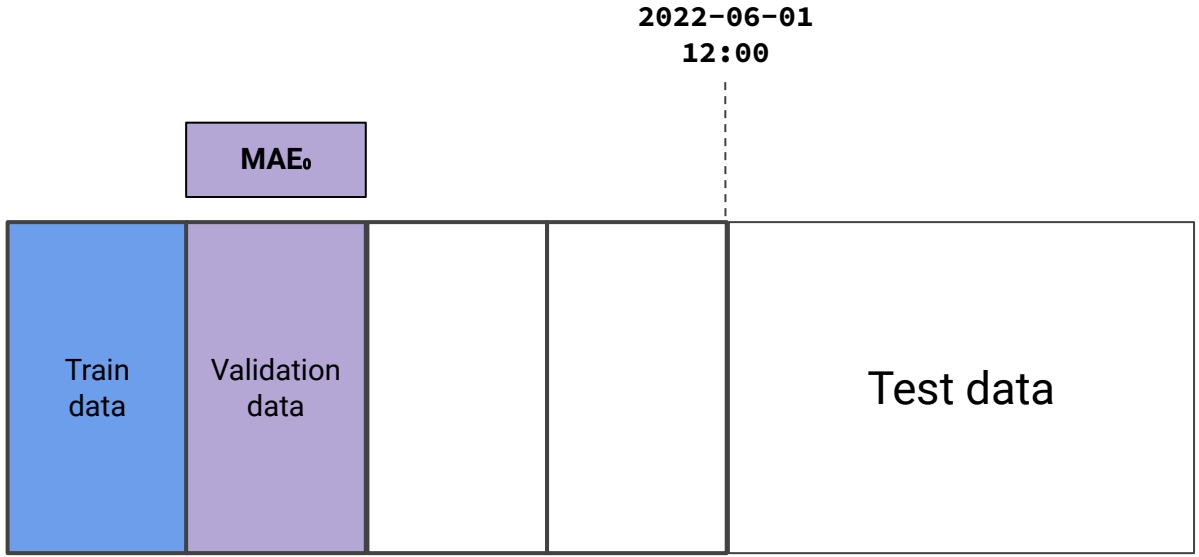
```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```



N split for better results

1. **Set hyper-parameters**

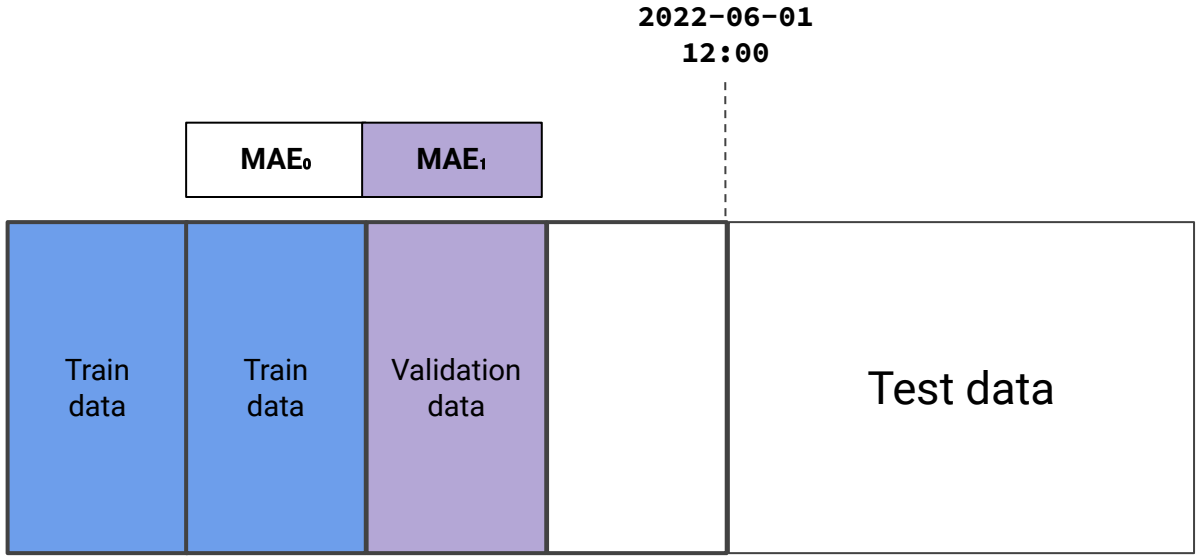
```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```



N split for better results

1. **Set hyper-parameters**

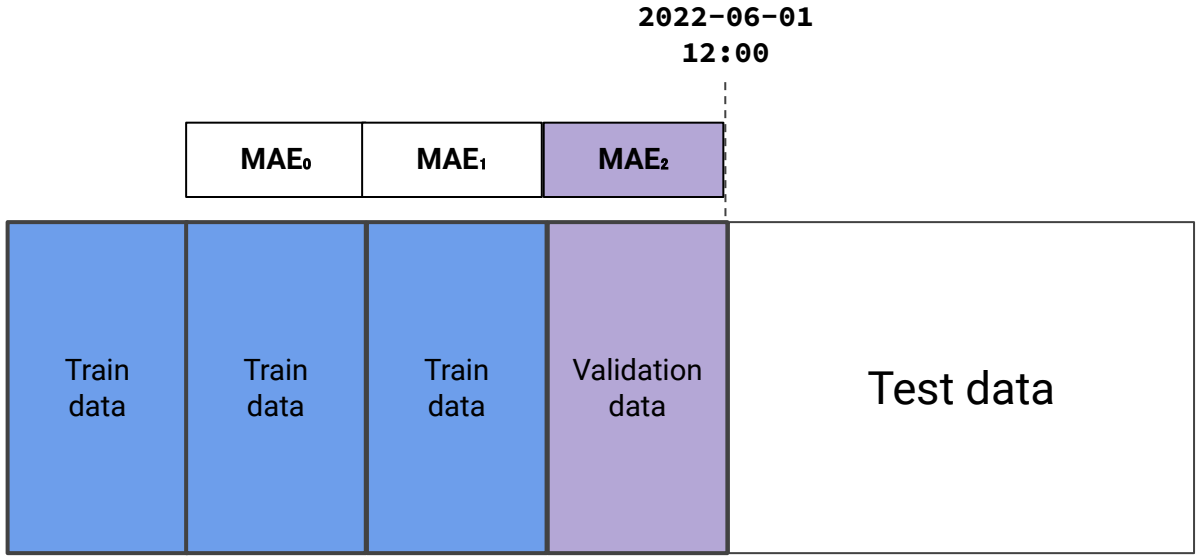
```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```



N split for better results

1. **Set hyper-parameters**

```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```



N split for better results

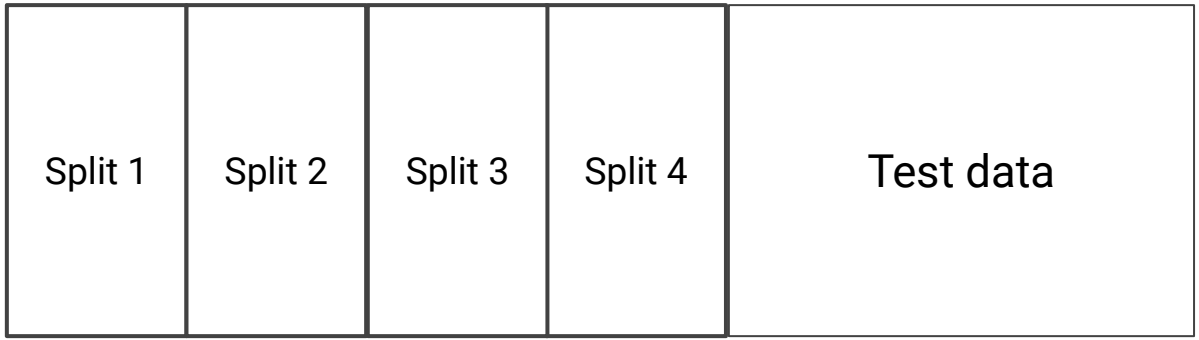
Suggest new set of hyper-parameters

2022-06-01 12:00

1. **Set hyper-parameters**

```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```

Average MAE_0, MAE_1 MAE_2



Hyperparameter search with Optuna

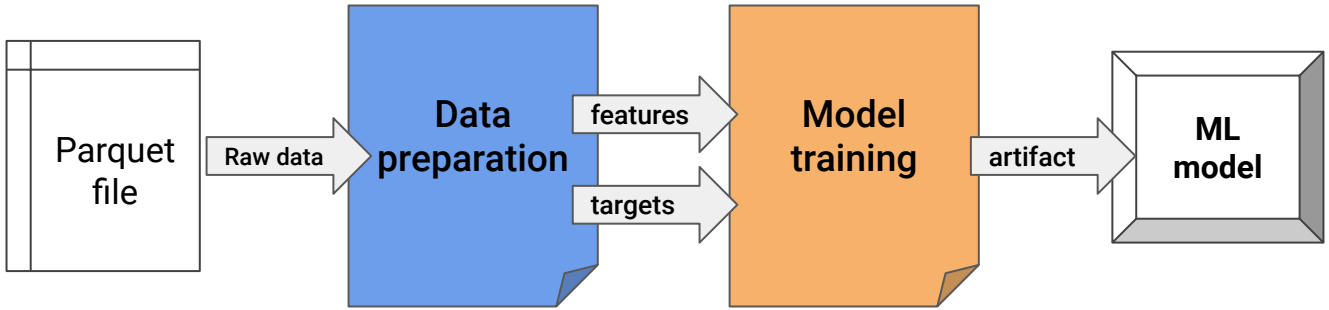
1. **Set hyper-parameters**

```
hyperparams = {
  "num_leaves": 45,
  "feature_fraction": 0.5,
  "bagging_fraction":0.75,
  "min_child_samples": 50,
}

model = LGBMRegressor(**hyperparams)
```

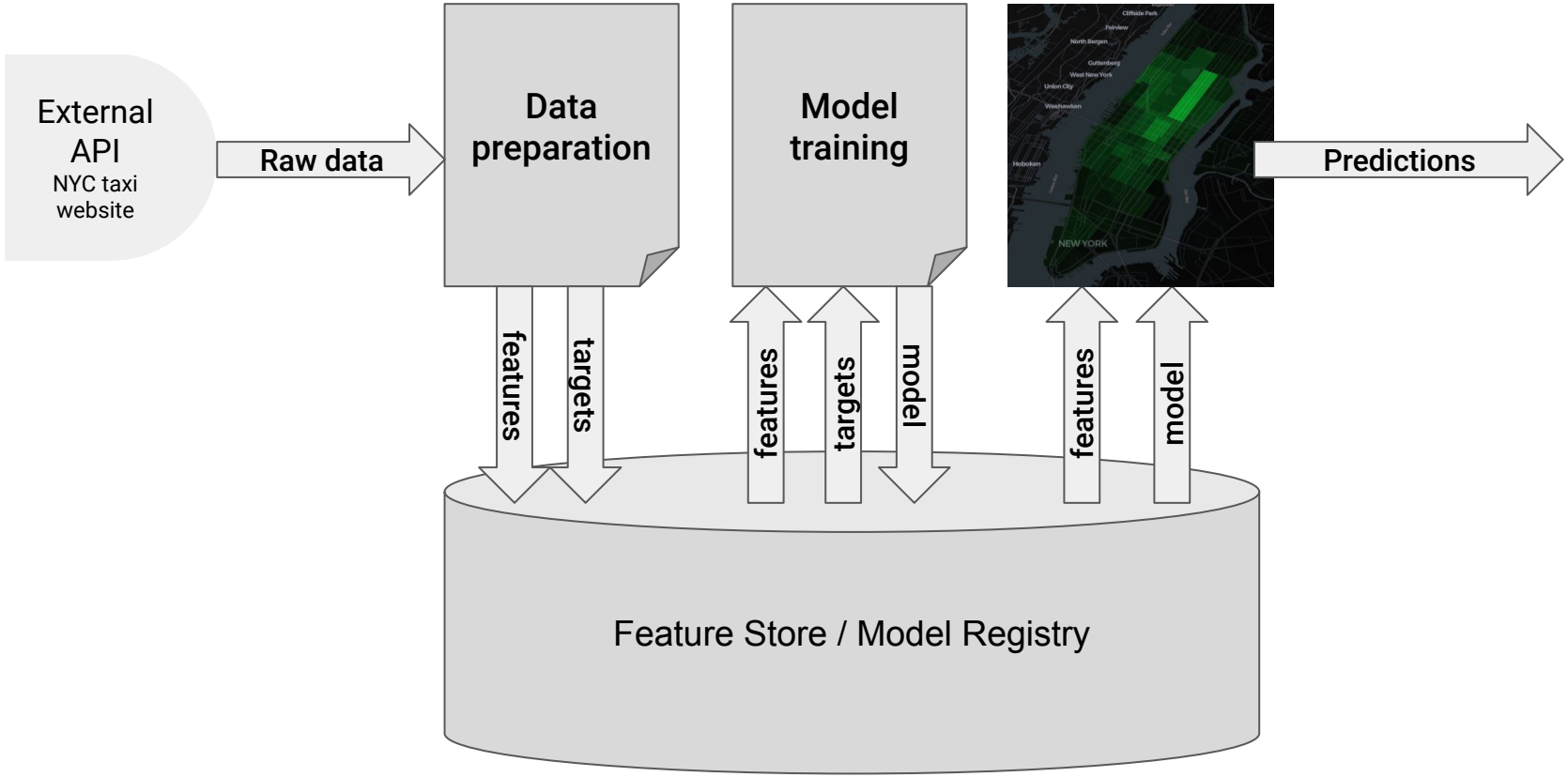
Validation MAE

4. MLOps

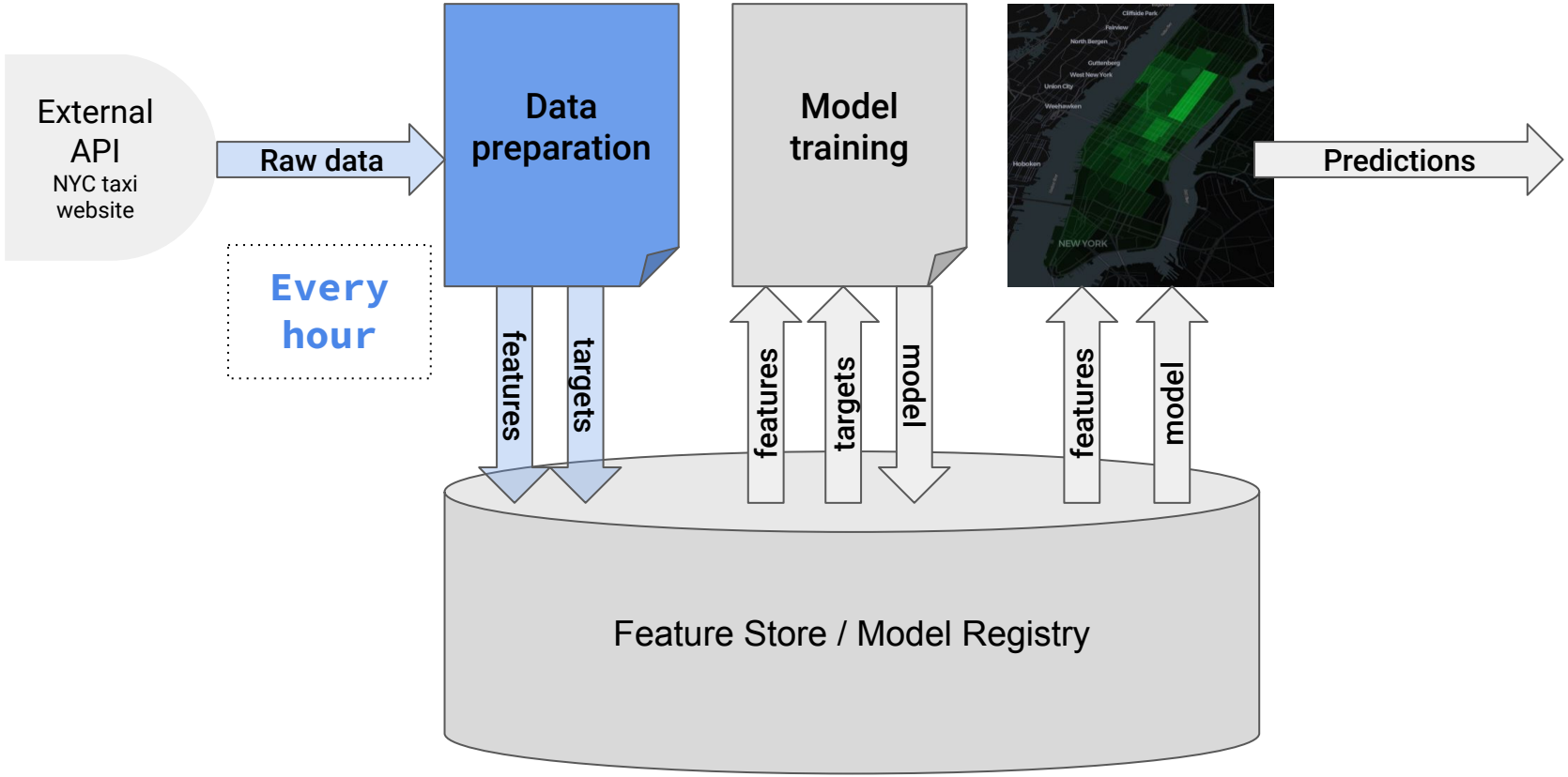
So far we did this...



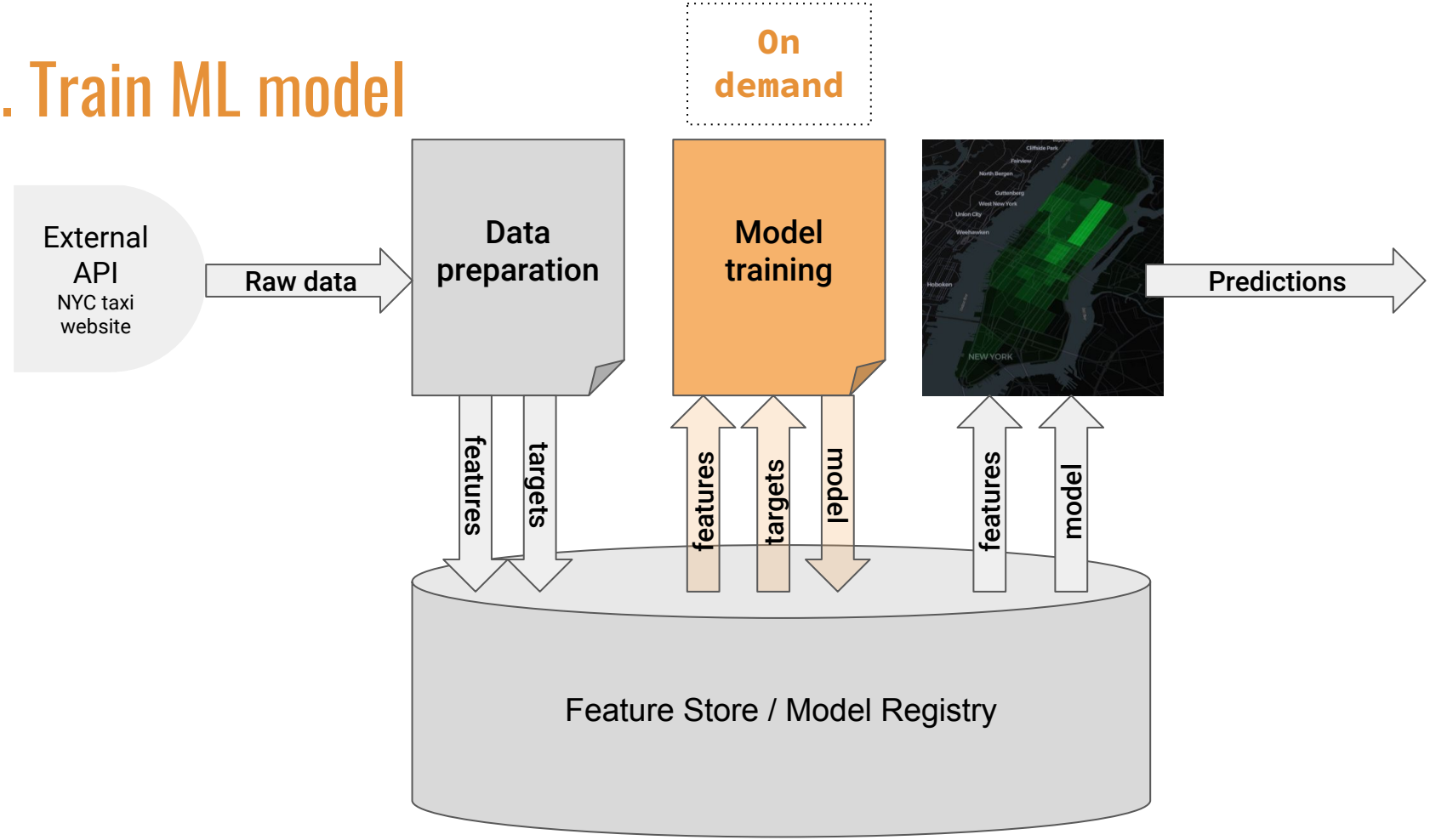
Batch-scoring system



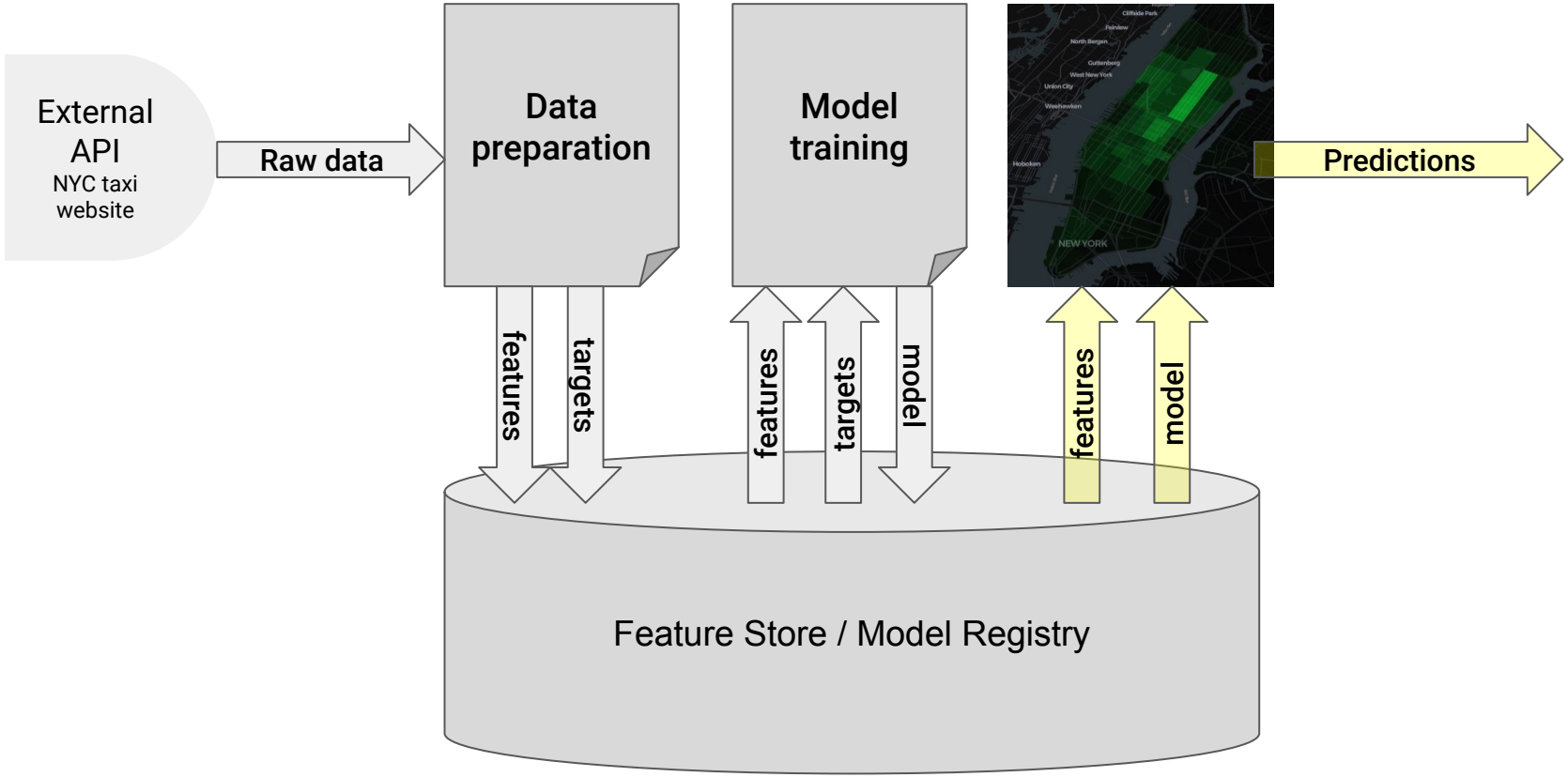
1. Prepare data



2. Train ML model



3. Generate predictions on recent data

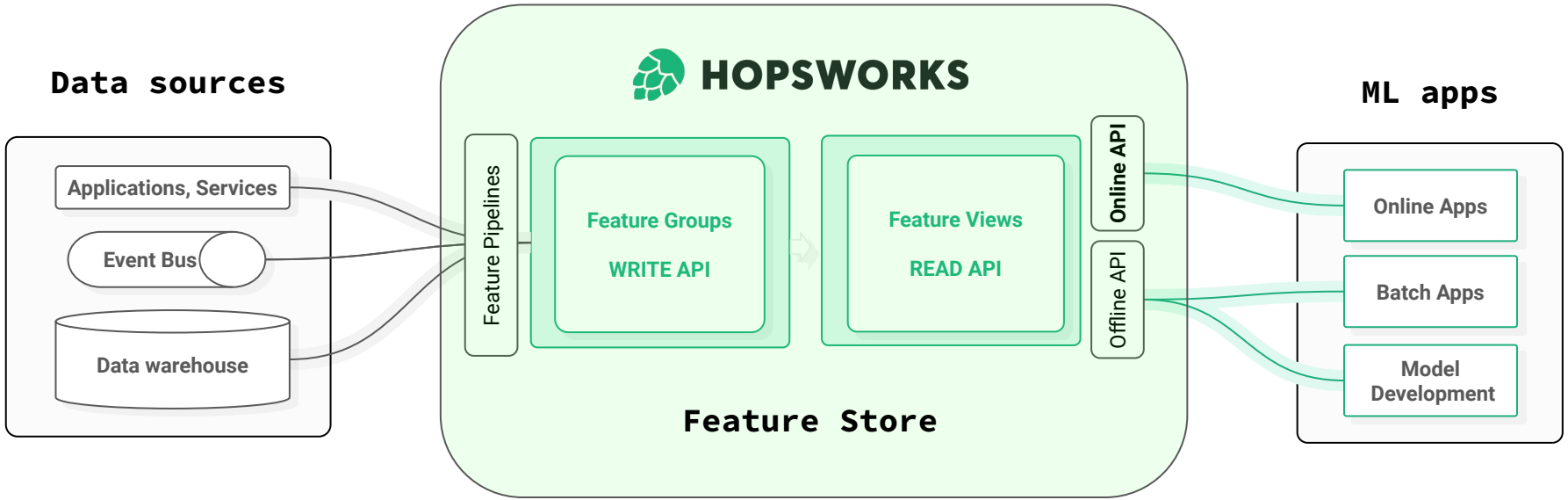


Serverless MLOps tools 🛠️

- **Hopsworks** as our Feature Store
- **GitHub Actions** to schedule and run jobs

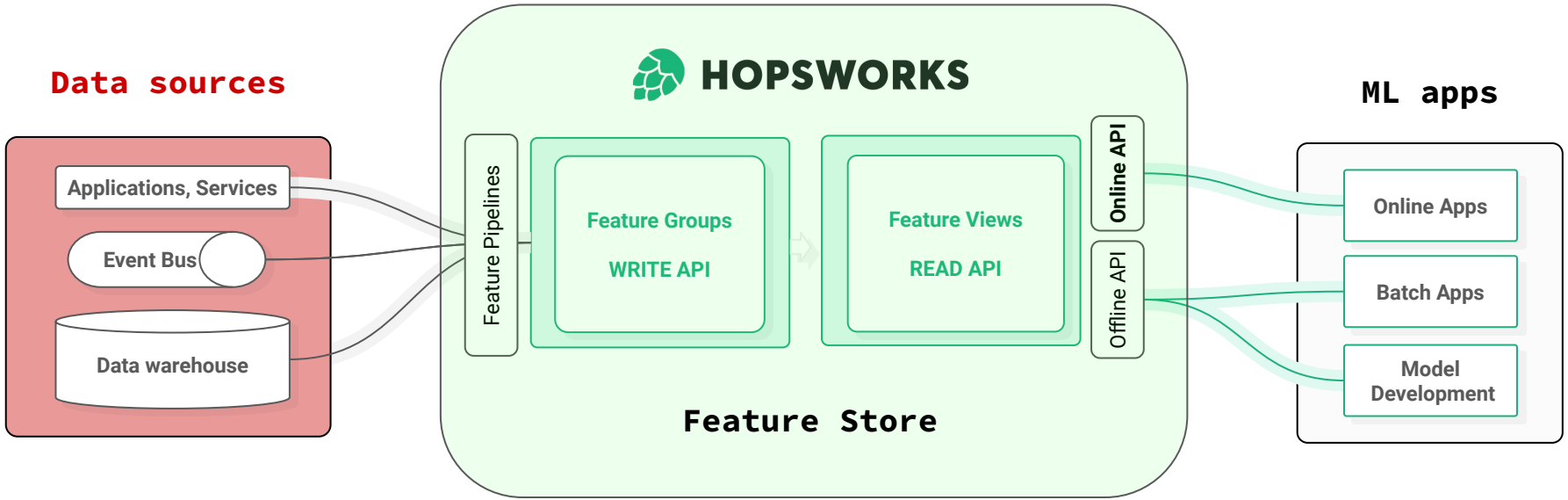
What’s a Feature Store?

Feature store



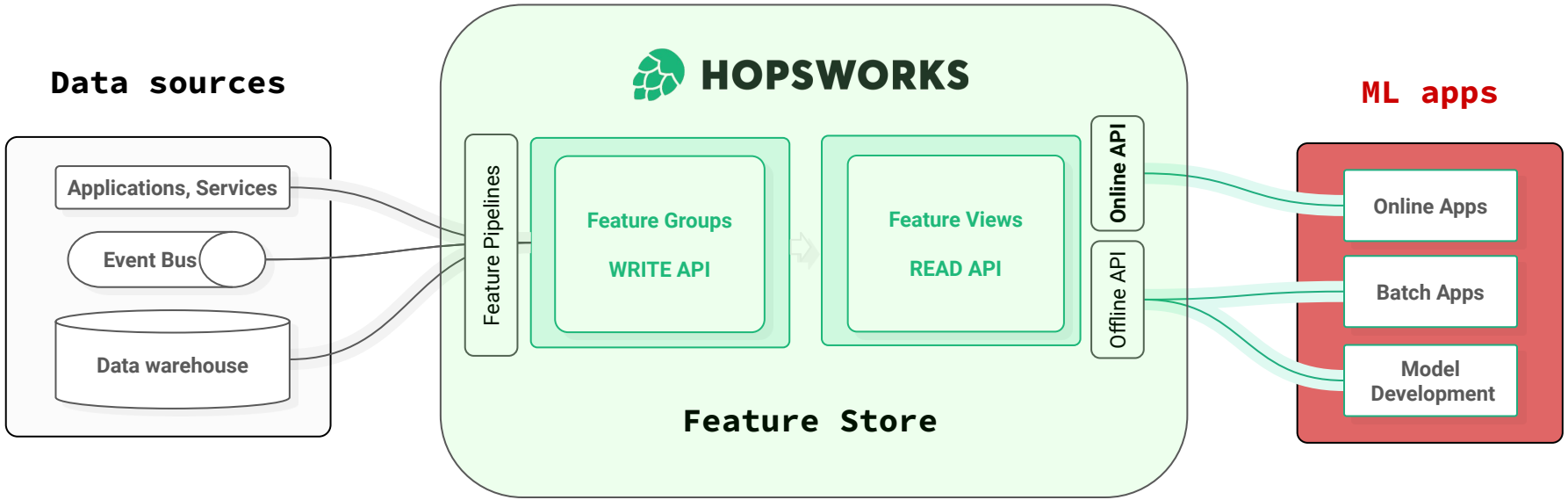
Source: https://www.serverless-ml.org

Feature store



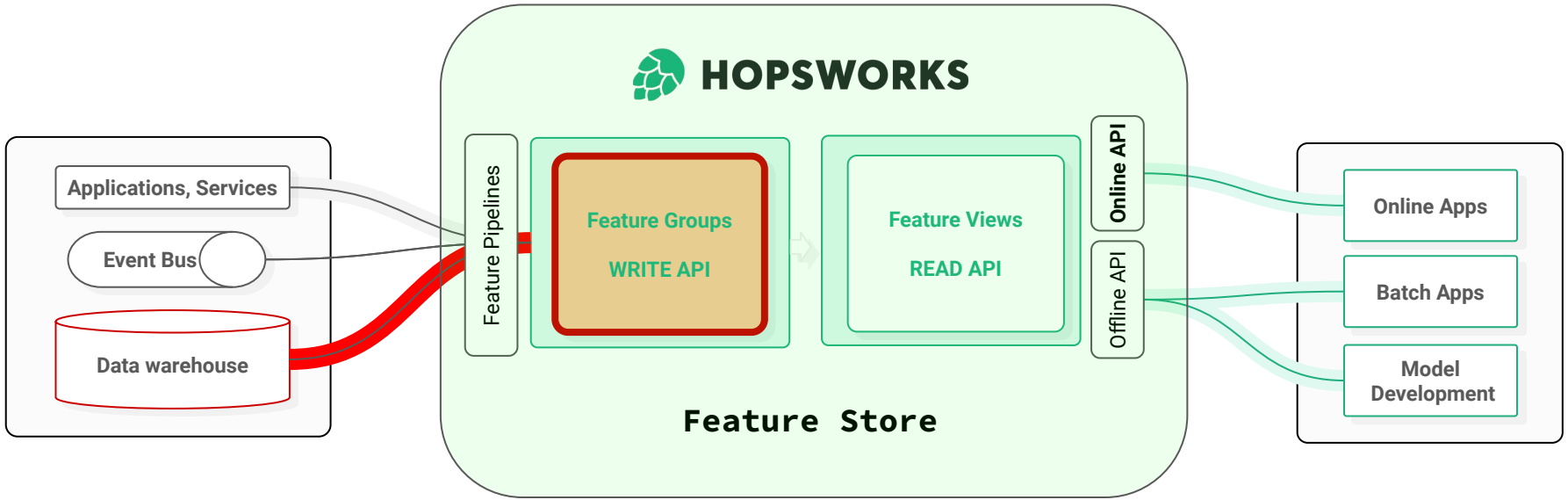
Source: https://www.serverless-ml.org

Feature store



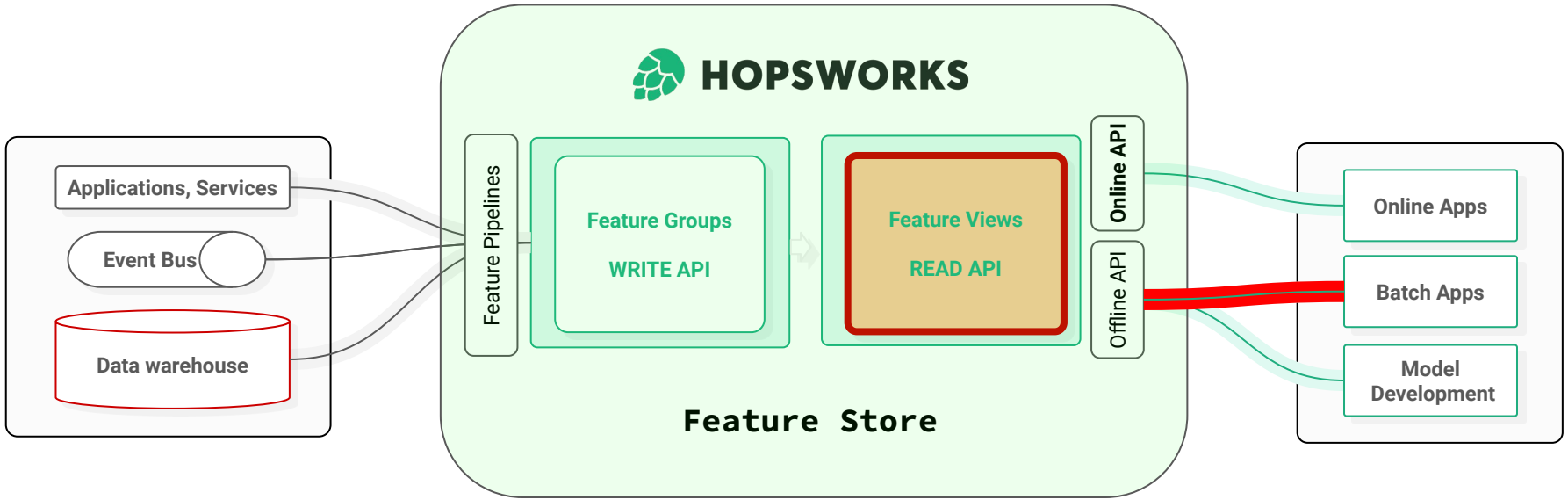
Source: https://www.serverless-ml.org

Feature store



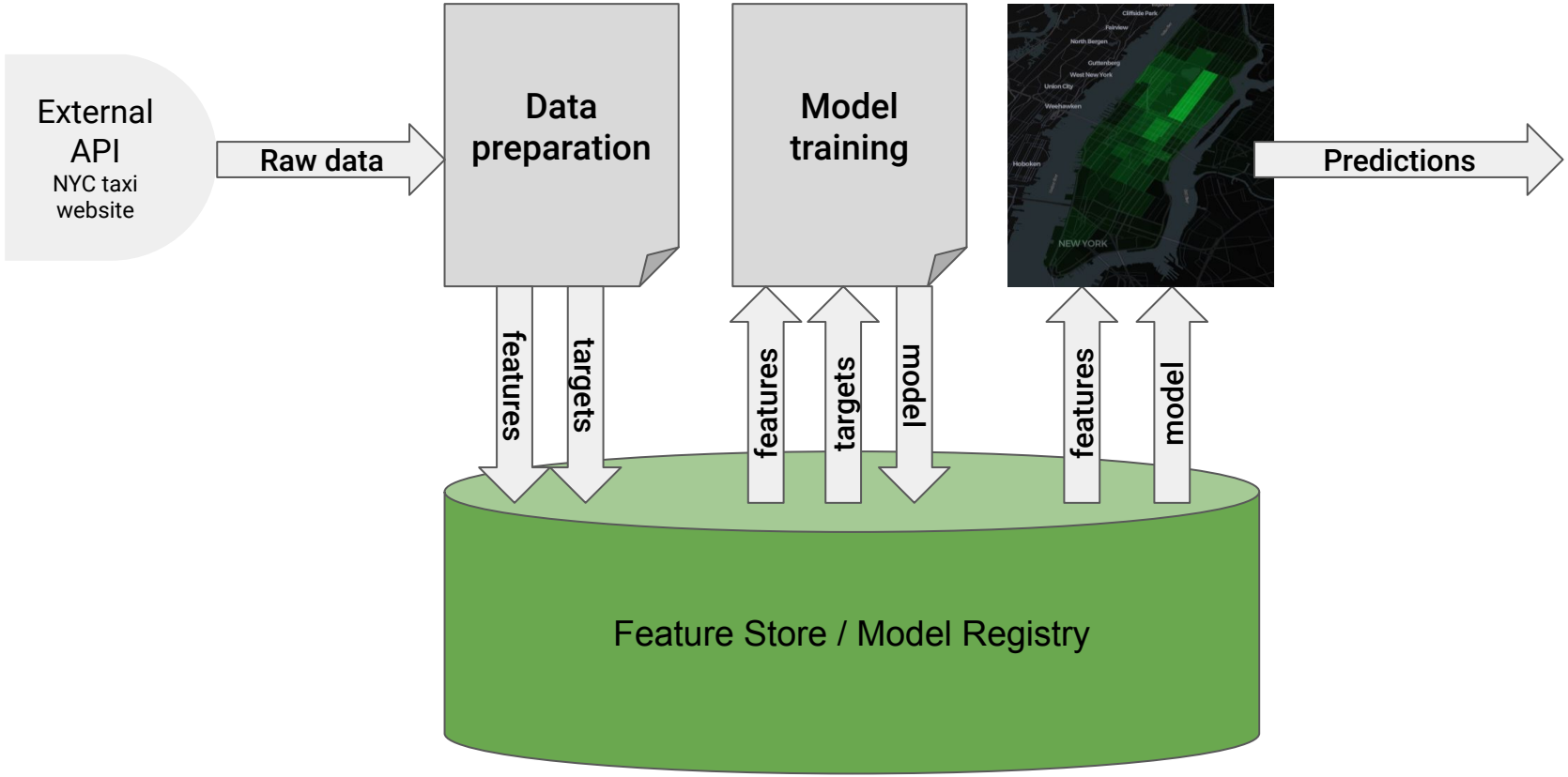
Source: https://www.serverless-ml.org

Feature store



Source: https://www.serverless-ml.org

Feature Store



Backfill the Feature Store