Machine Learning Pipelines

Rayid Ghani and Kit Rodolfa

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT

Heinz College
INFORMATION SYSTEMS • PUBLIC POLICY • MANAGEMENT

Things to remember

- **Due Monday:** Features and Pipeline v0 assignment

Finishing Up Temporal Validation

Parameters

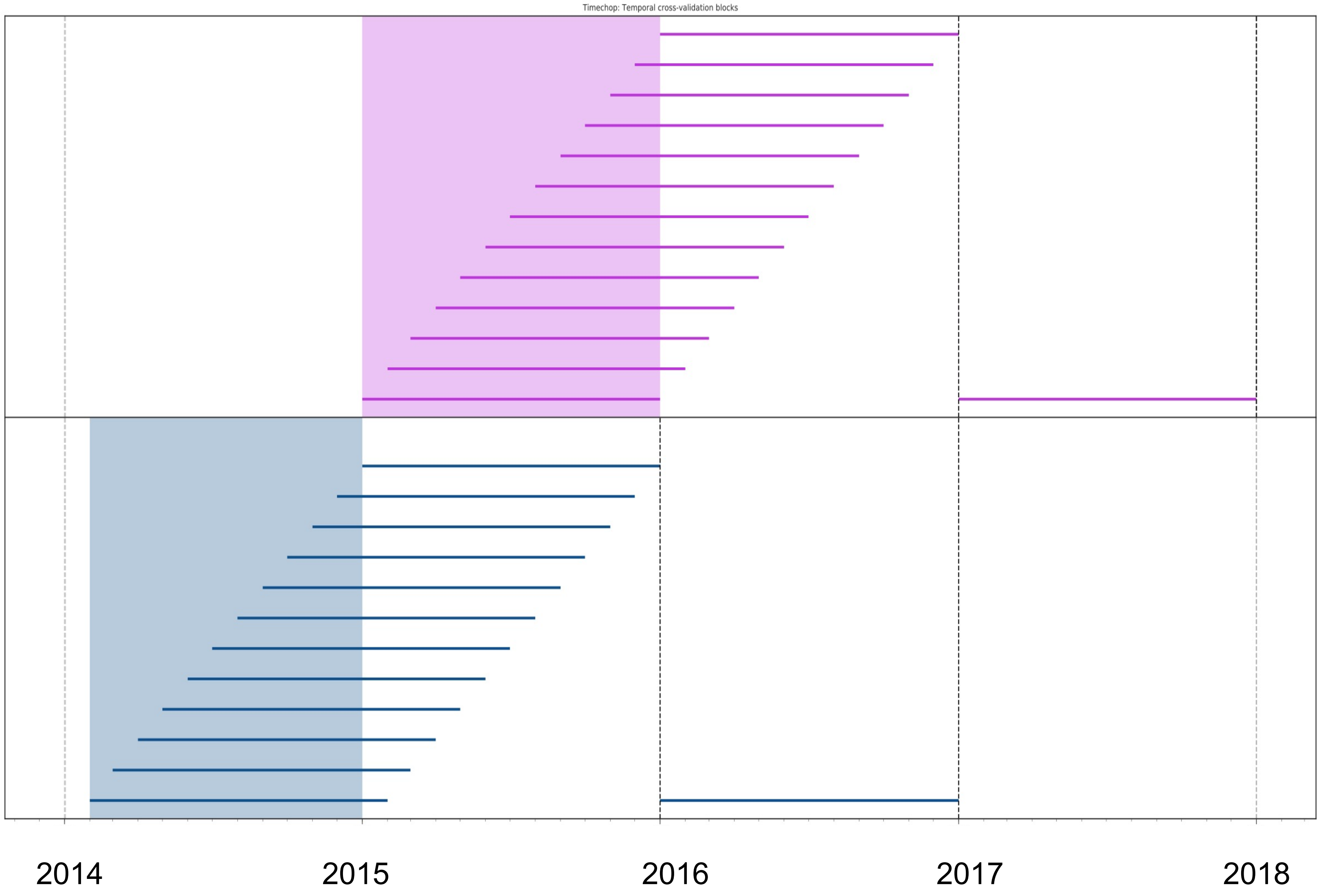
- How far back to go when training models? (max training history)
  - To the beginning of time (expanding training window)?
  - Fixed history (rolling training window)?
  - Something else?
  - How far back do you get your features from?
- How much to move forward from train-validation pair 1 to train-validation pair 2?
  - A day?
  - A month?
  - Something else?

Other considerations

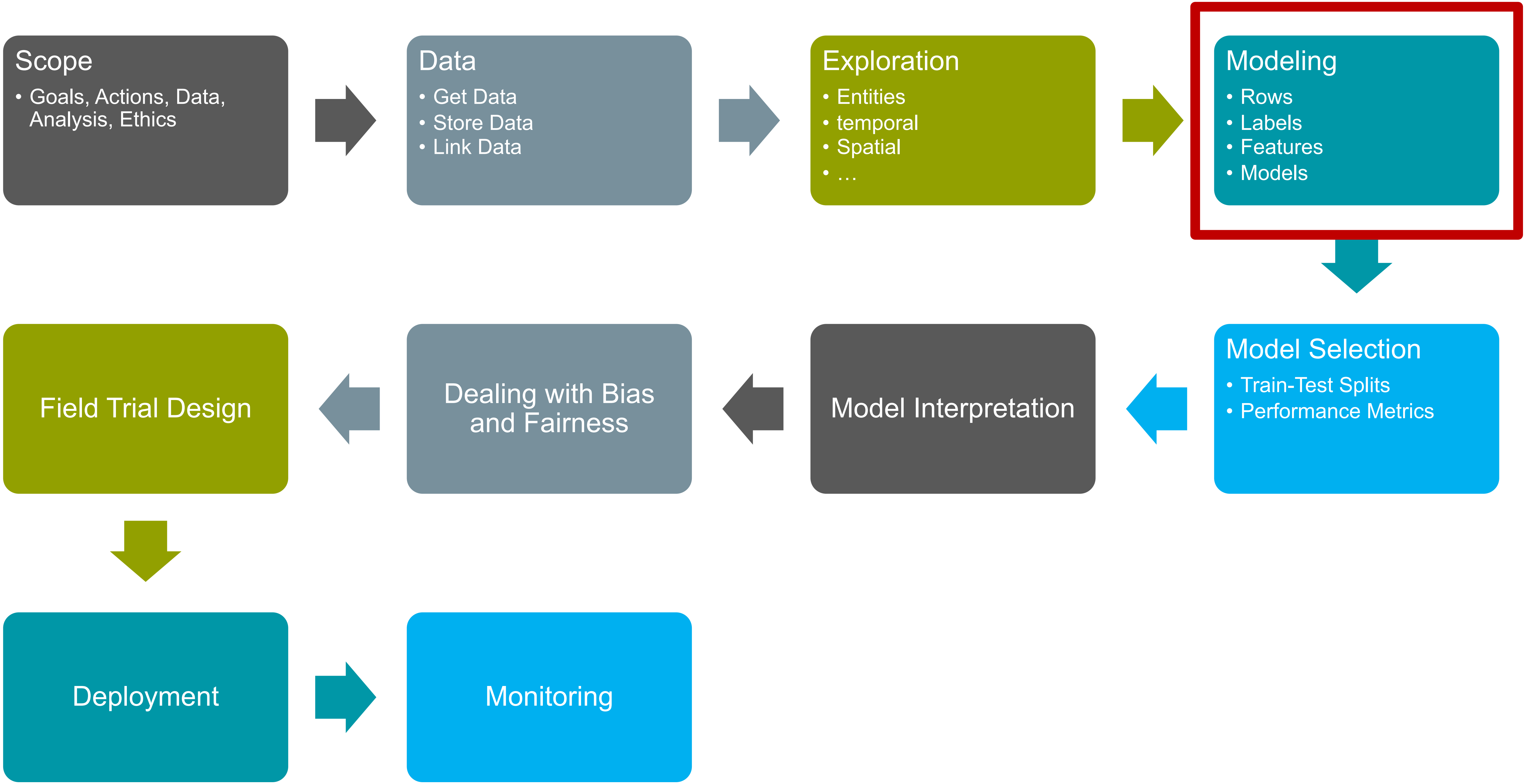
- If making repeated predictions about the same entity at different times, how often should an entity be repeated in the training data?
    - In an event-based deployment setup?
    - In a “take action at regular-ish intervals” deployment?
- What about in the validation set?

Temporal configuration parameters

```yaml
temporal_config:
    feature_start_time: '2014-01-01'
    feature_end_time: '2018-01-01'
    label_start_time: '2014-01-02'
    label_end_time: '2018-01-01'

    model_update_frequency: '1y'
    training_label_timespans: ['1y']
    training_as_of_date_frequencies: '1month'

    test_durations: '0d'
    test_label_timespans: ['1y']
    test_as_of_date_frequencies: '1month'

    max_training_histories: '1y'
```



Machine Learning Pipelines


Scope
• Goals, Actions, Data, Analysis, Ethics
Data
• Get Data
• Store Data
• Link Data
Exploration
• Entities
• temporal
• Spatial
• …
Modeling
• Rows
• Labels
• Features
• Models
Model Selection
• Train-Test Splits
• Performance Metrics
Model Interpretation
Dealing with Bias and Fairness
Field Trial Design
Deployment
Monitoring

Things we will cover

- What is an ML Pipeline?
- Why should we build ML pipelines?
- What components should it have?
- Best Practices
- Good Examples

What is an ML Pipeline?

- Supports end-to-end workflow for an ML project/system
- Modular
- Reconfigurable

Why build a pipeline?

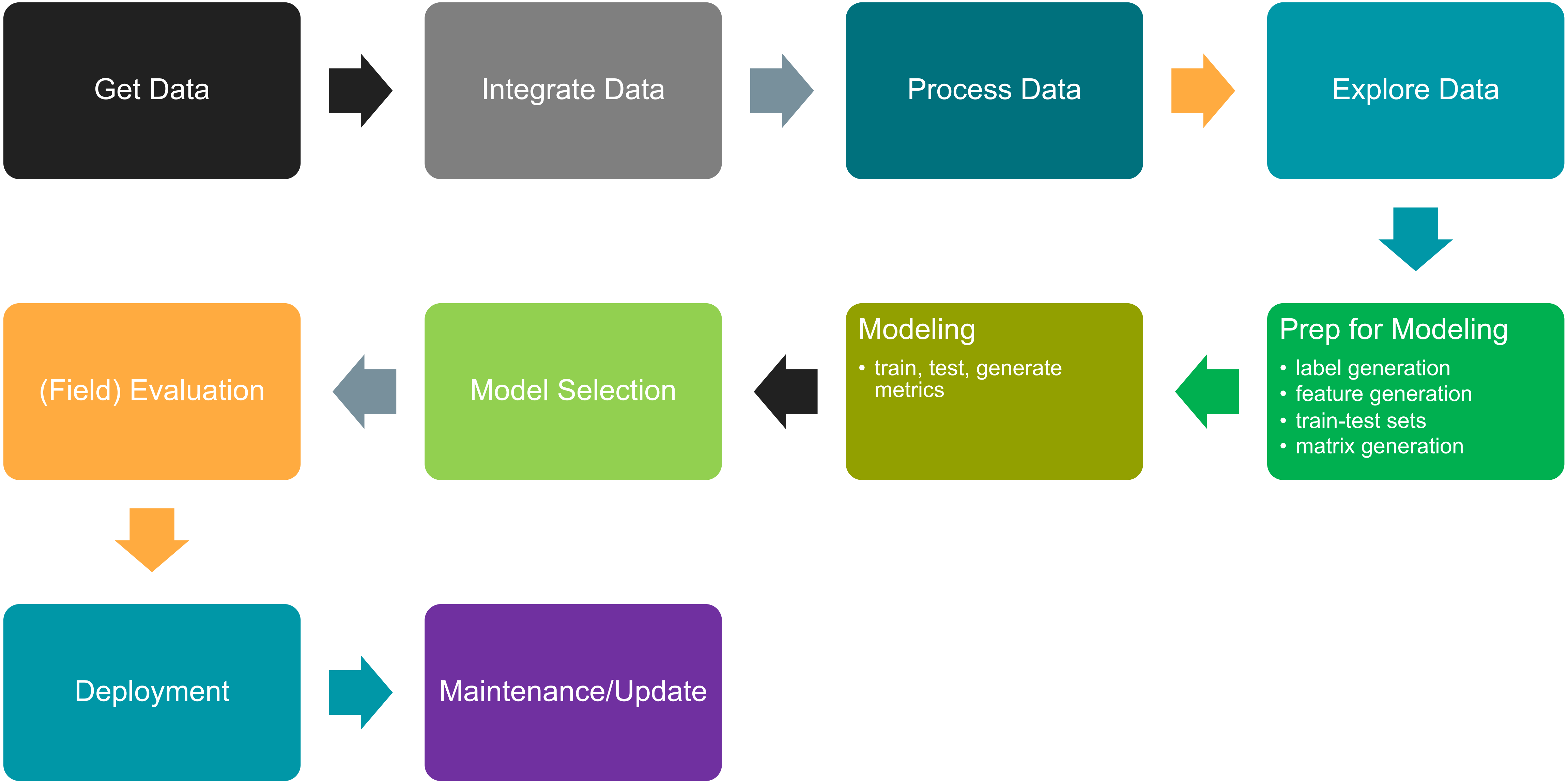
- Reusable across projects
- Test new ideas, components, hypothesis easily
- Reduce bugs/errors
- Allows reproducibility of analysis and results

What makes a pipeline?

- Inputs
- Components
- (Intermediate and final) outputs

Pipeline Flow & Components

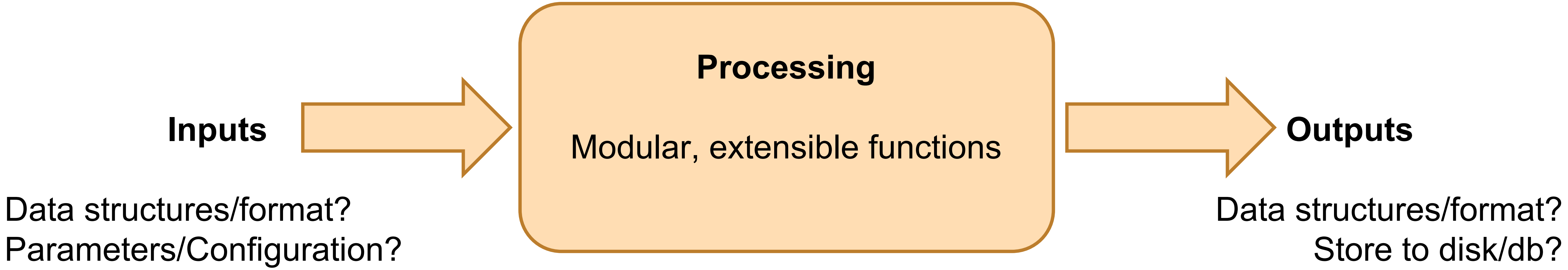
Pipeline Flow



What components does a pipeline have?

- Read/Load Data (from csv, db, api)
- Integrate Data (dedupe, link)
- Process Data (cleaning)
- Explore Data (descriptive stats, correlations, outliers, over time, clustering)
- Modeling Prep
  - Create training and test sets
  - Missing values (fill/impute, create dummy)
  - Transformations (scale/normalize, log, square, root)
  - Feature Generation
  - Label Generation
  - Define metric(s)
- Modeling
  - Build model(s) on training sets
  - Apply model(s) on test sets
  - Calculate metric(s)
- Model Selection
- Field Trial
- Deploy
- Maintain

Things to keep in mind about each component



Components: Data Acquisition & Integration

- Get Data
  - API, CSV, Database
- Store Data
  - Database
- Integrate Data
  - Record Linkage

Components: Explore and Prepare data

- Data Exploration
  - Distributions
  - Missing Values
  - Correlations
  - Other Patterns
- Pre-Processing
  - Leakage
  - Deal with Missing values
  - Scaling
  - Data errors

Components: Feature Creation

- Data comes with fields or columns (if it’s even structured), not features
- Common Features
  - Discretization
  - Transformations
  - Interactions/Conjunctions
  - Disaggregation
  - Aggregations
    - Temporal
    - Spatial
- How are you handling imputation of missing values?

Components: Method Selection

- Select pool of methods applicable for task
- For loop over a large number of methods
  - For loop over parameters

Components: Validation

- Using historical data
  - Methodology
  - Metric

- Field Experiment
  - Methodology
  - Metric

Deployment

- Model monitoring
- Re-training
  - How often?
  - Re-select methods?
- Scoring

What types of variations do you want to test using your pipeline?

- Different models
- Model parameters
- Different Labels/Outcomes
- Different Deployment Settings
- Different Feature (Groups)
- Different Metrics

Best Practices

- Draw a diagram of the pipeline:
  - What function runs each step? What are the inputs? What are the outputs?
- Config files (yaml, json, py)
- Make each step modular and extensible so it can easily be re-used
- Build a **simple**, end-to-end version first, then add more functionality
- Think about how you'll store outputs:
  - Store models as pickles
  - Store predictions in databases
  - Store evaluation metrics in databases
  - Database schema to store everything in

- Timesplitter
  - Input: start time, end time, update time, prediction time
  - Output: pairs of <train start time, train end time, test start time, test end time>
- CohortCreator
  - Input: timesplitter output, cohort definition,[entity_ids, as of date]
  - Output: cohort matrix <entity_id, as_of_date>
- LabelCreator
  - Input: pairs <entity_id, as_of_date>, label definition
  - Output: matrix <entity_id, as_of_date, label>
- FeatureCreator
  - Input: pairs <entity_id, as_of_date>, feature definition(s)
  - Output: matrix <entity_id, as_of_date, feature(s)>
- ModelTrainer
  - Input: model definition, matrix, feature columns, label column
  - Output: model object (stored), model definition
- ModelScorer
  - Input: model object, matrix, feature columns
  - prediction scores
- Evaluator
  - Prediction scores, label column, metric(s)

Progression

1. Determine Input/outputs for each component
2. Example of code for each component
3. python file that imports each component and builds a pipeline for 1 train test set, 1 model, 1 metric, etc.
4. Loop over additional variations
5. Move parameters from python file to external config file
6. SQL and python

Things to remember

- **Due Friday:** Project Proposal

- Remainder of this week:
  - Wednesday: tech session on using triage for ML pipelines
  - Time for group meetings/project work on Thursday

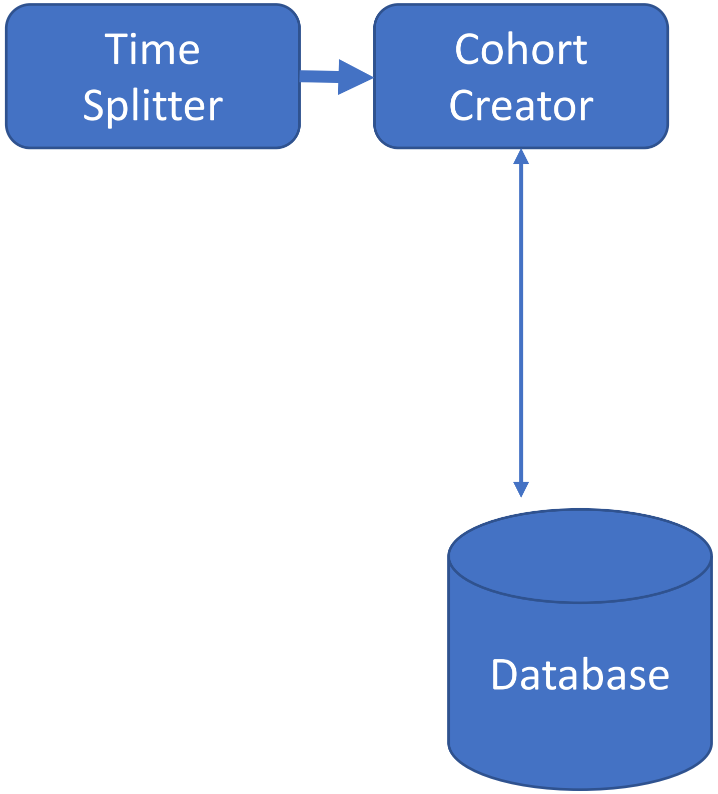
- Coming up next week:
  - Weekly review (before class on Tuesday)
  - Transductive Top-k Reading (for Tuesday)
  - Wednesday tech sessions: Python + SQL
  - Due Friday: Proposal peer reviews

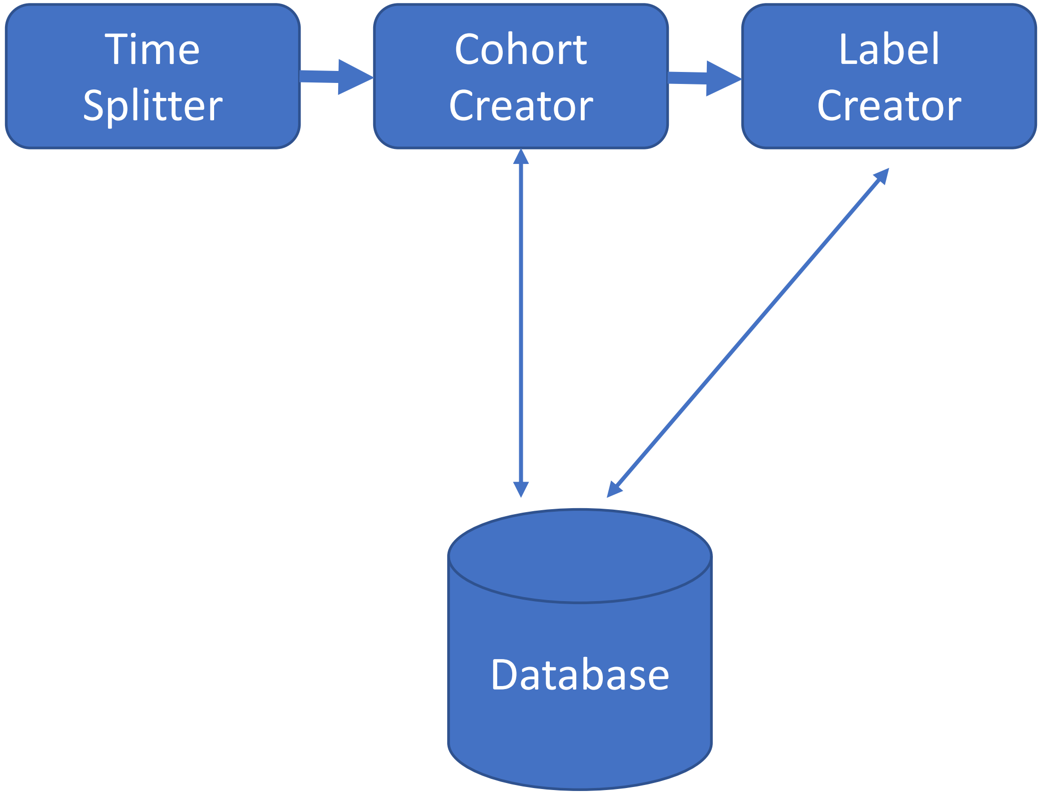
Appendix:
Slides from Previous Tech Session

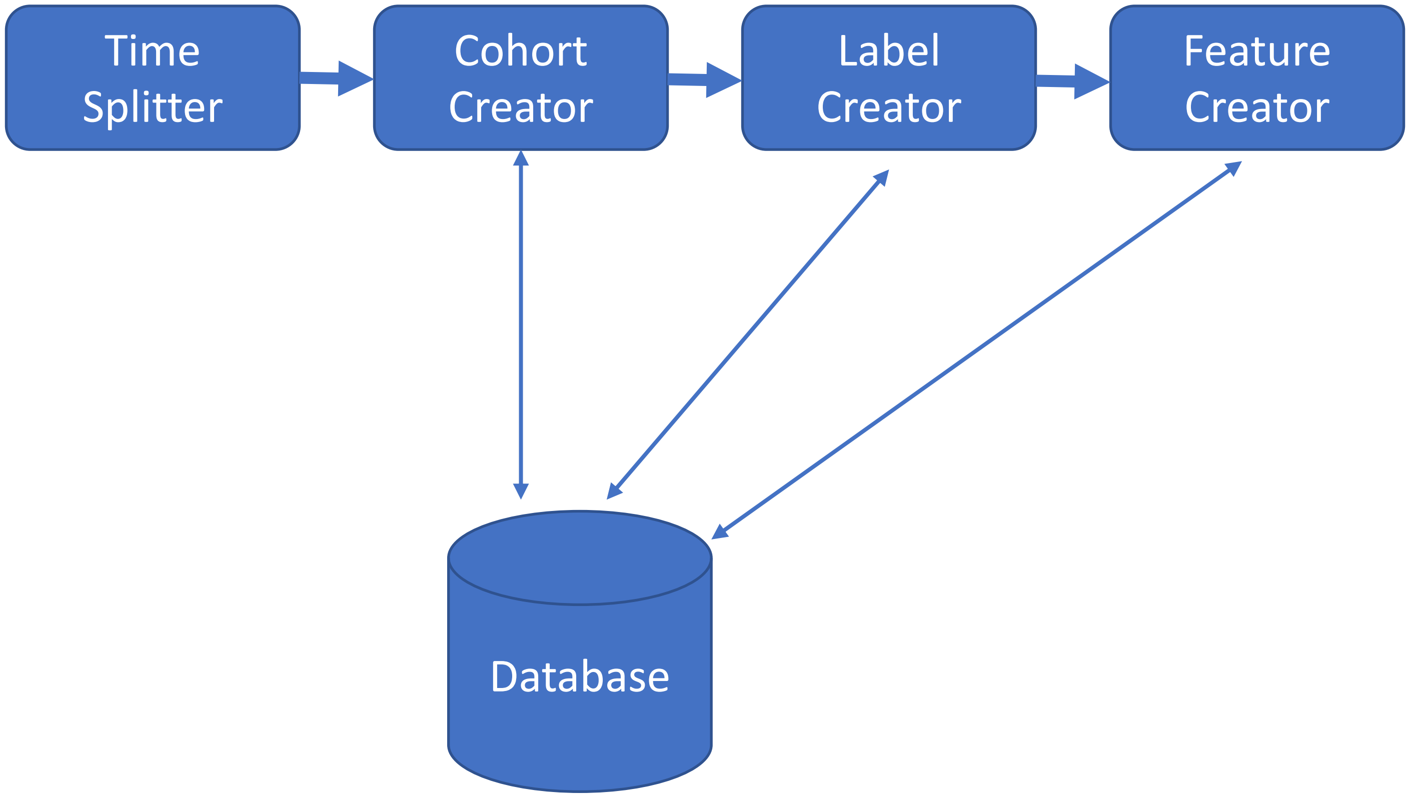

Train Examples
Labels
Validate Examples
Labels


Features Data
Train Examples
Labels
Validate Examples
Labels

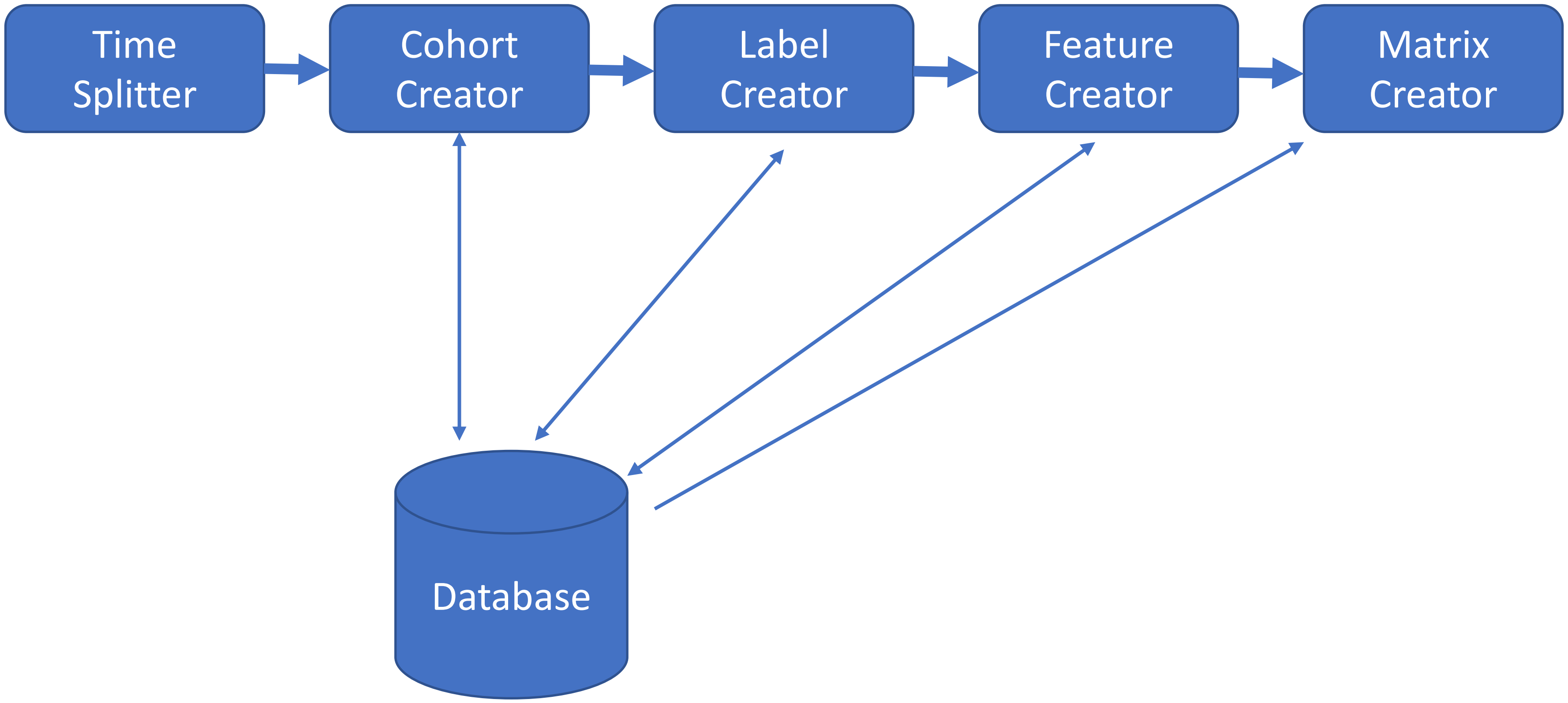
Time Splitter

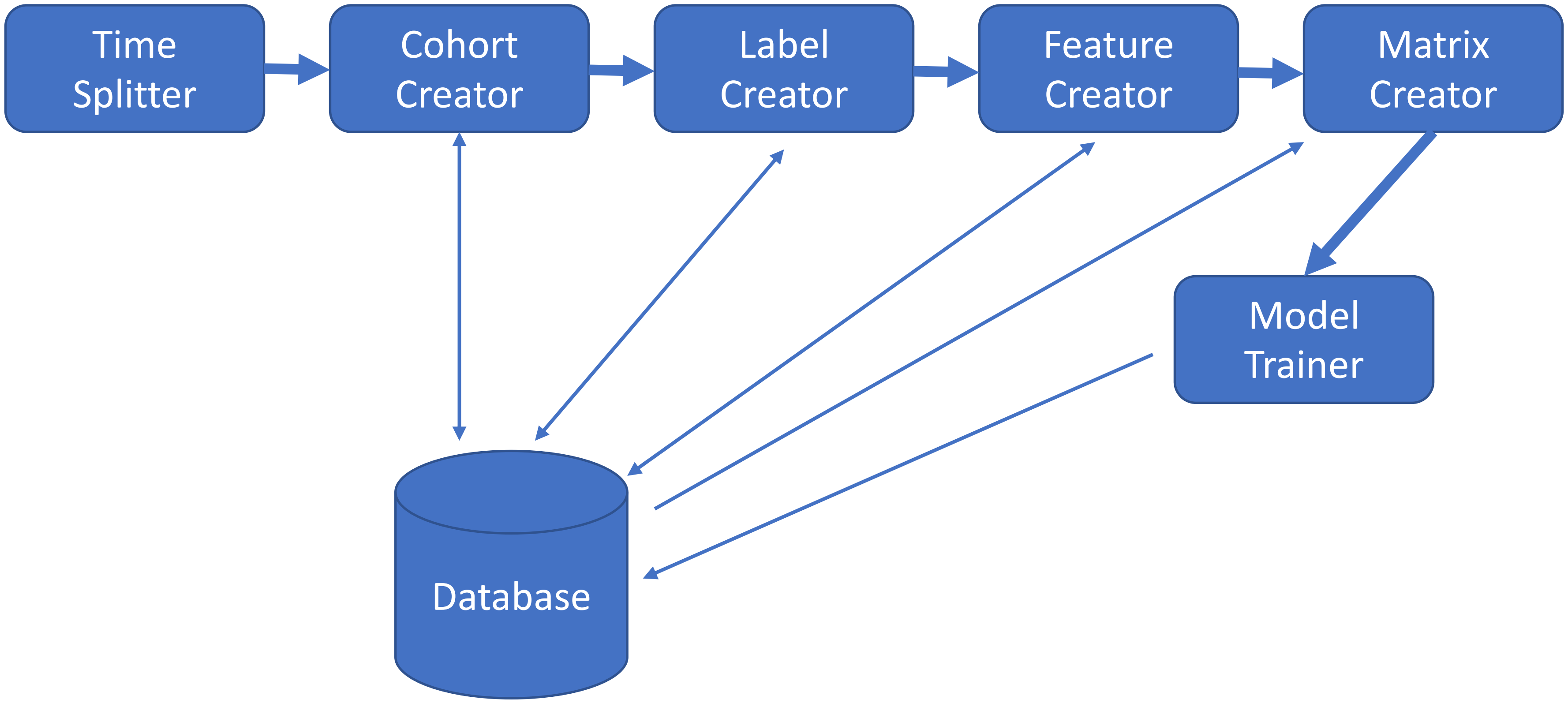

Time Splitter
Cohort Creator
Database

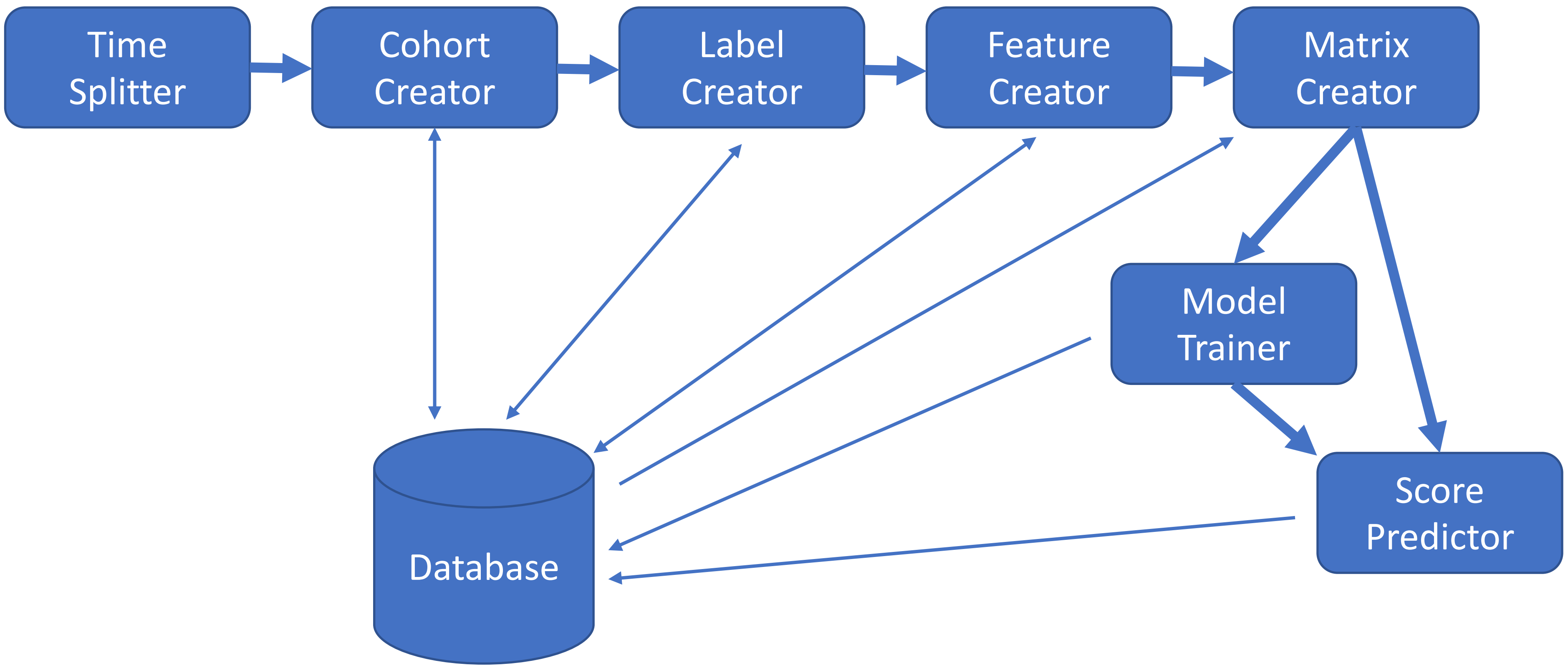

Time Splitter
Cohort Creator
Label Creator
Database

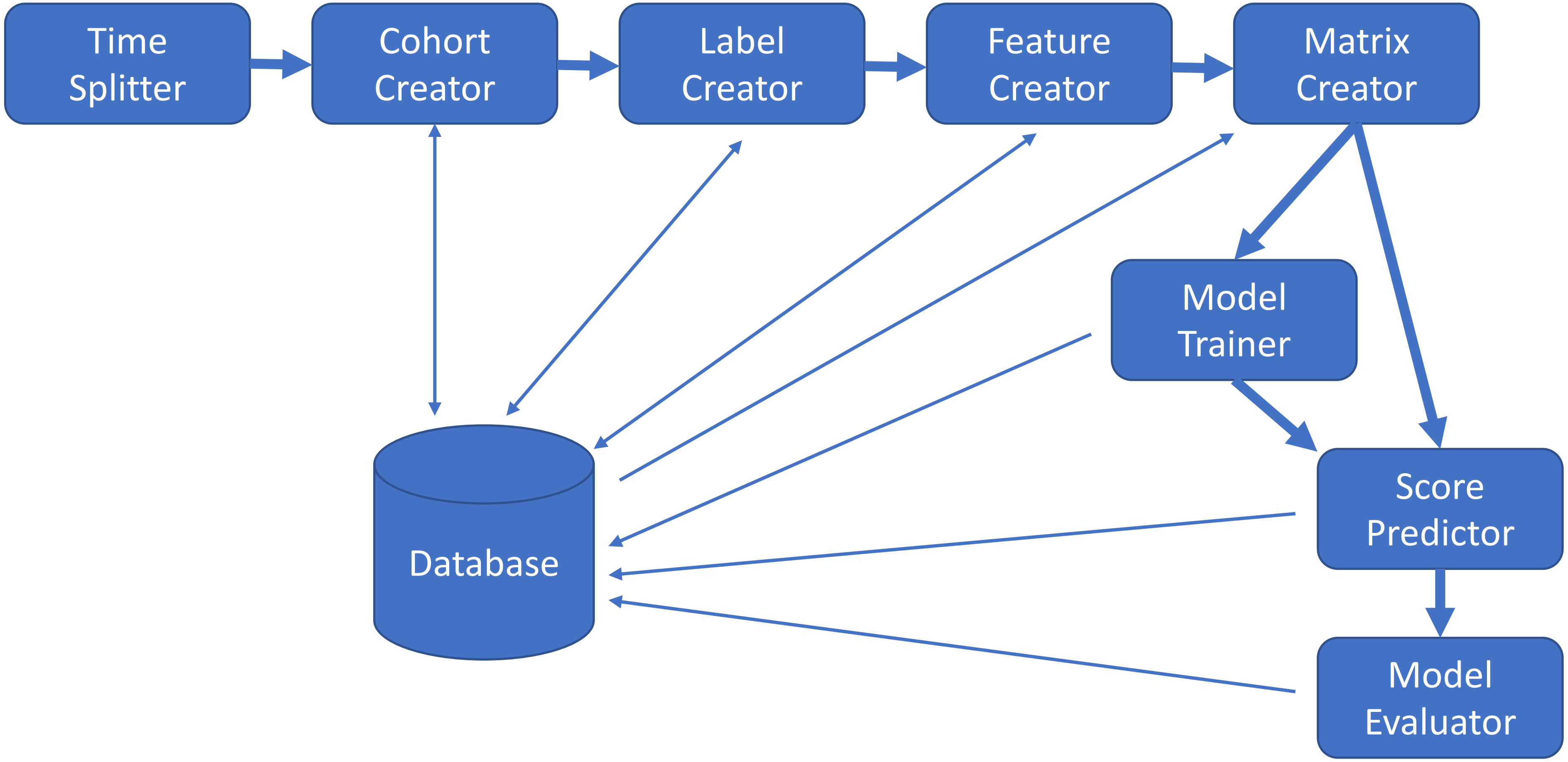


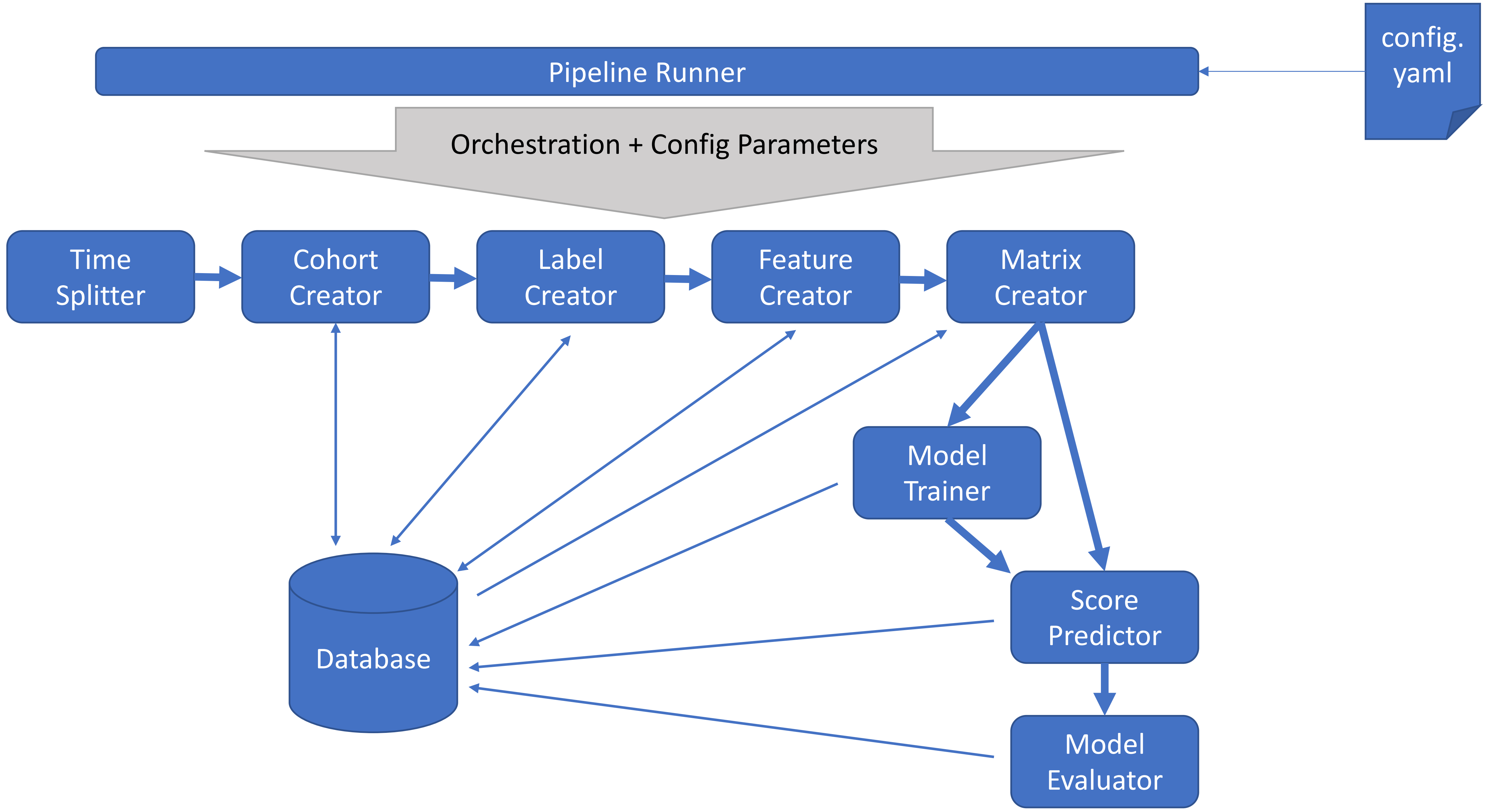
Time Splitter
Cohort Creator
Label Creator
Feature Creator
Database

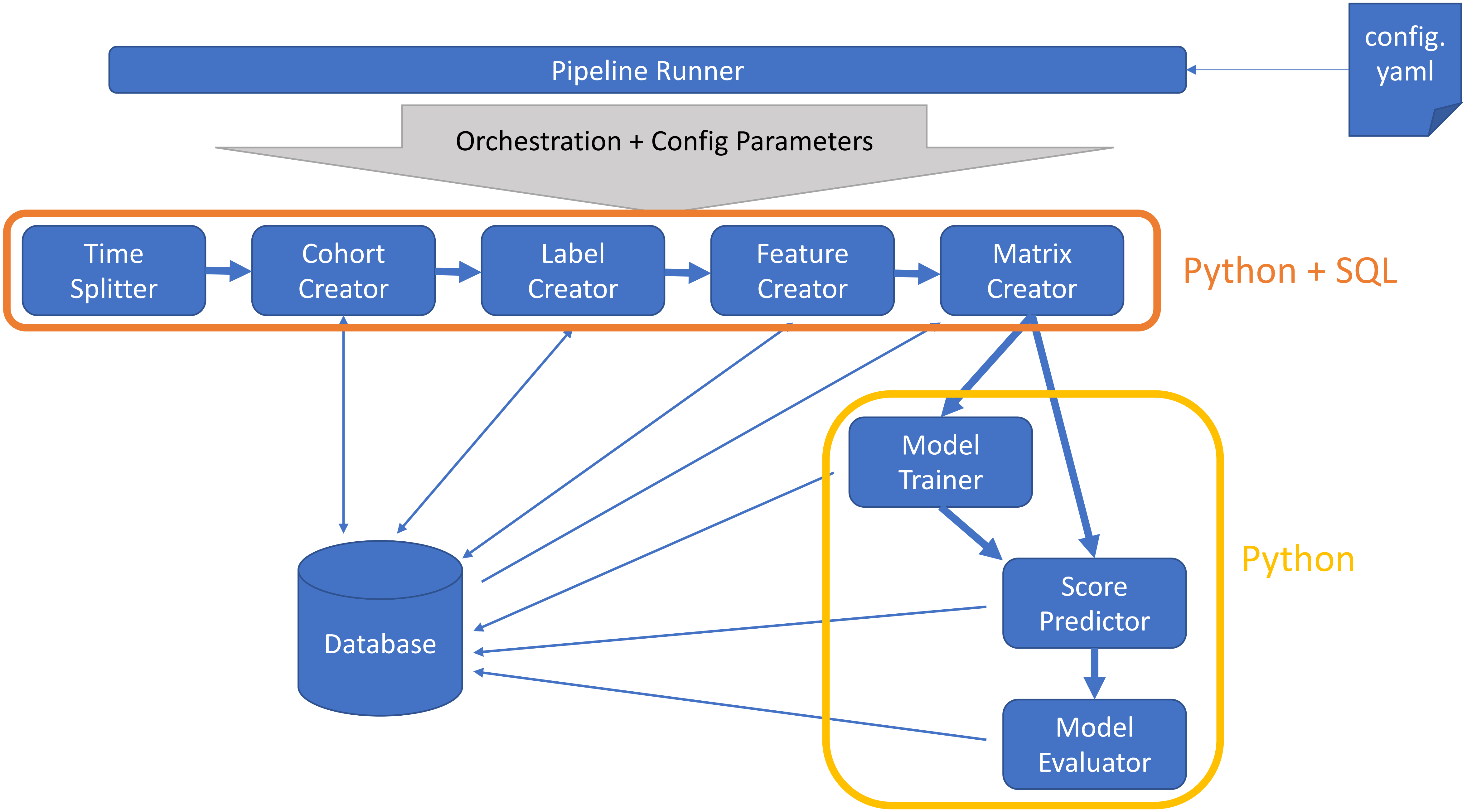

Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Database


Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Database

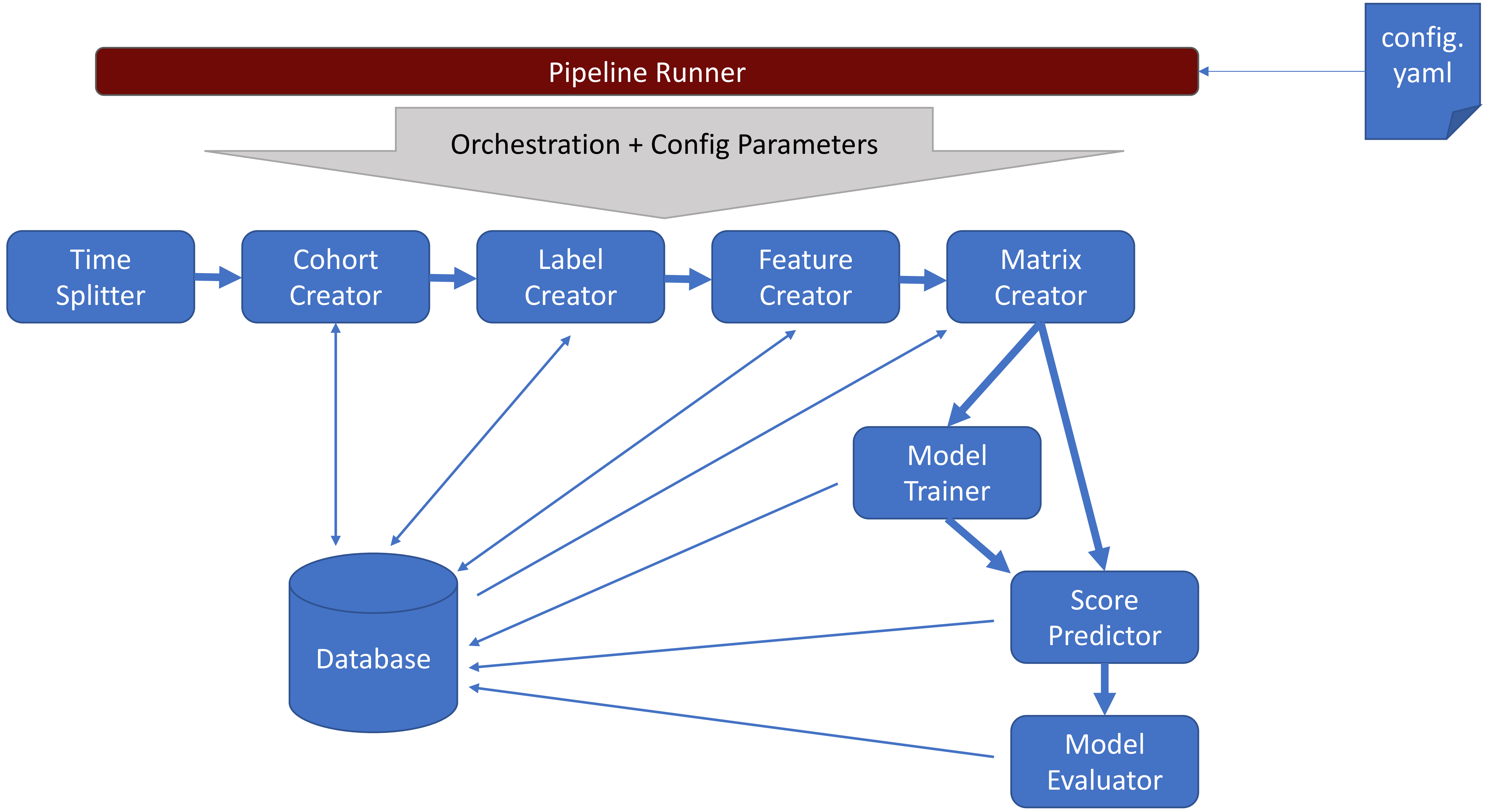

Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Score Predictor
Database


Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Score Predictor
Model Evaluator
Database


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Score Predictor
Model Evaluator
Database

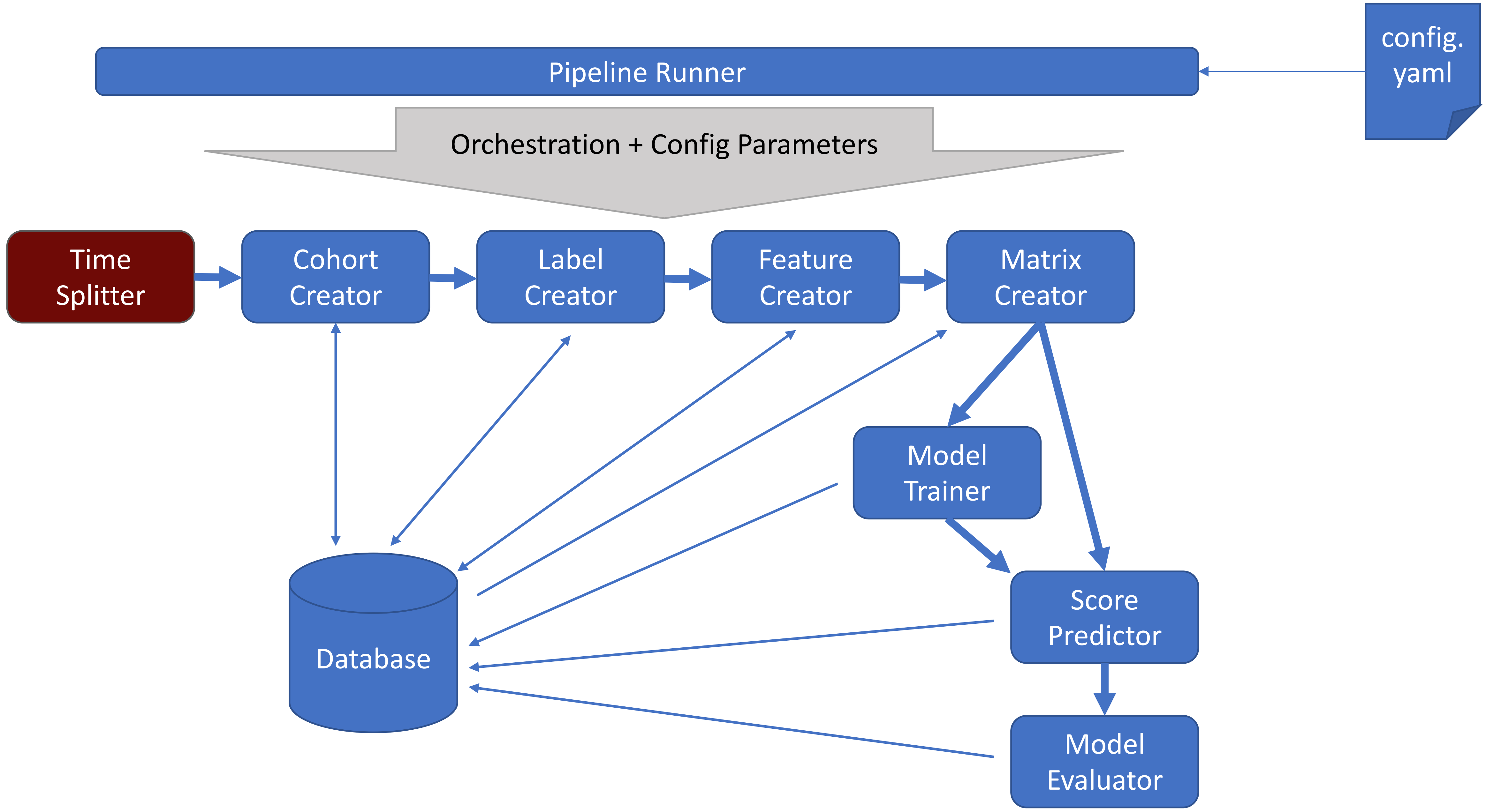


Pipeline Runner
config.
yaml
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Python + SQL
Model
Trainer
Score
Predictor
Model
Evaluator
Python
Database


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

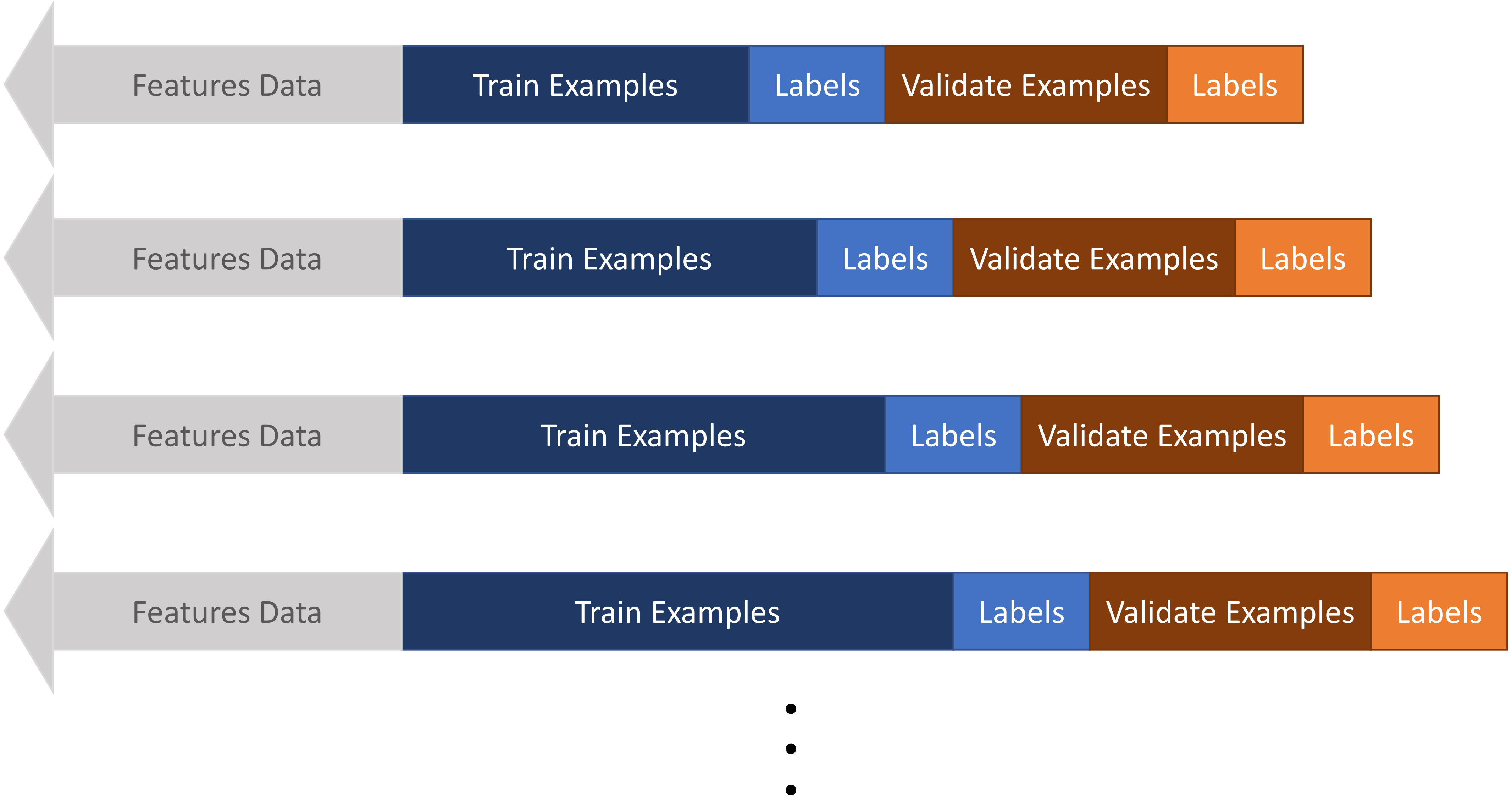
Pipeline Runner

```python
from time_splitter import split_time
from data_prep import (
    cohort_creator, label_generator,
    feature_creator, matrix_maker
)
from modeling import (
    expand_model_grid, train_model,
    predict_scores, evaluate_model
)
import sys
import yaml

def run(config):
    splits = split_time(config['temporal'])
    cohort_table = cohort_creator(splits, config['cohort'])
    label_table = label_generator(cohort_table, config['label'])
    feature_table = feature_creator(cohort_table, config['features'])

    for split in splits:
        train_matrix, validate_matrix = matrix_maker(
            split, cohort_table, label_table, feature_table
        )
        model_grid = expand_model_grid(config['model_grid'])
        for model_params in model_grid:
            model = train_model(model_params, train_matrix)
            scores = predict_scores(model, validate_matrix)
            metrics = evaluate_model(scores, validate_matrix)

if __name__ == '__main__':
    config_path = sys.argv[1]
    with open(config_path) as f:
        config = yaml.safe_load(f)
    run(config)
```


config. yaml
Pipeline Runner
Orchestration + Config Parameters
Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Database
Score Predictor
Model Evaluator

Time Splitter

Time Splitter



Time Splitter

INPUTS

Temporal Parameters:

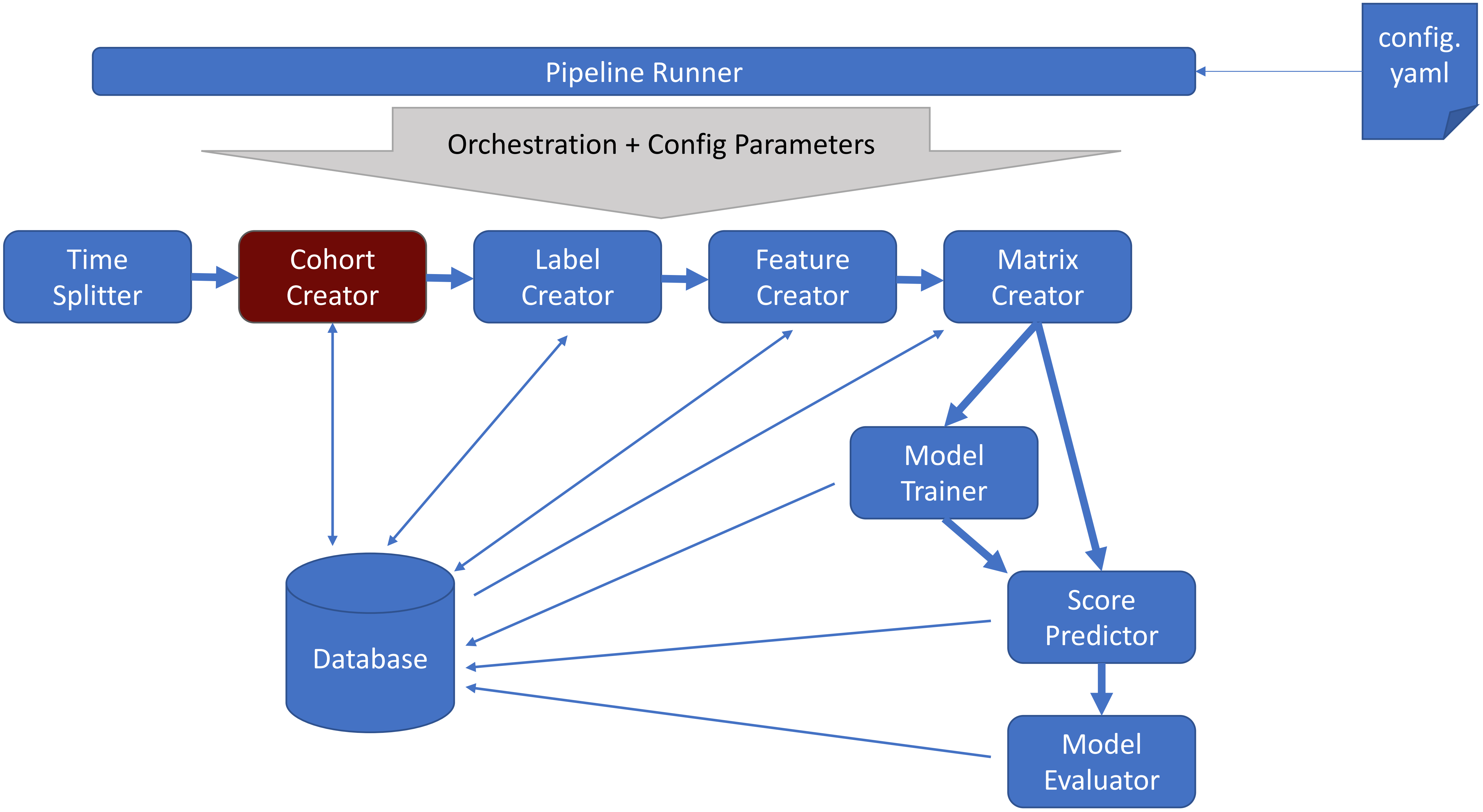
- Beginning of time
- End of time
- Label Window
- Example History
- Update Frequency

OUTPUTS

Paired train/validate ***sets of dates***:

```
 [
   (train_start_1, train_end_1),
   (test_start_1, test_end_1)
 ],
 [
   (train_start_2, train_end_2),
   (test_start_2, test_end_2)
 ], …
```

Note: **Not** splits of the **data**!
(you still need the full history for features)



config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Cohort Creator

INPUTS

Train/validate split dates

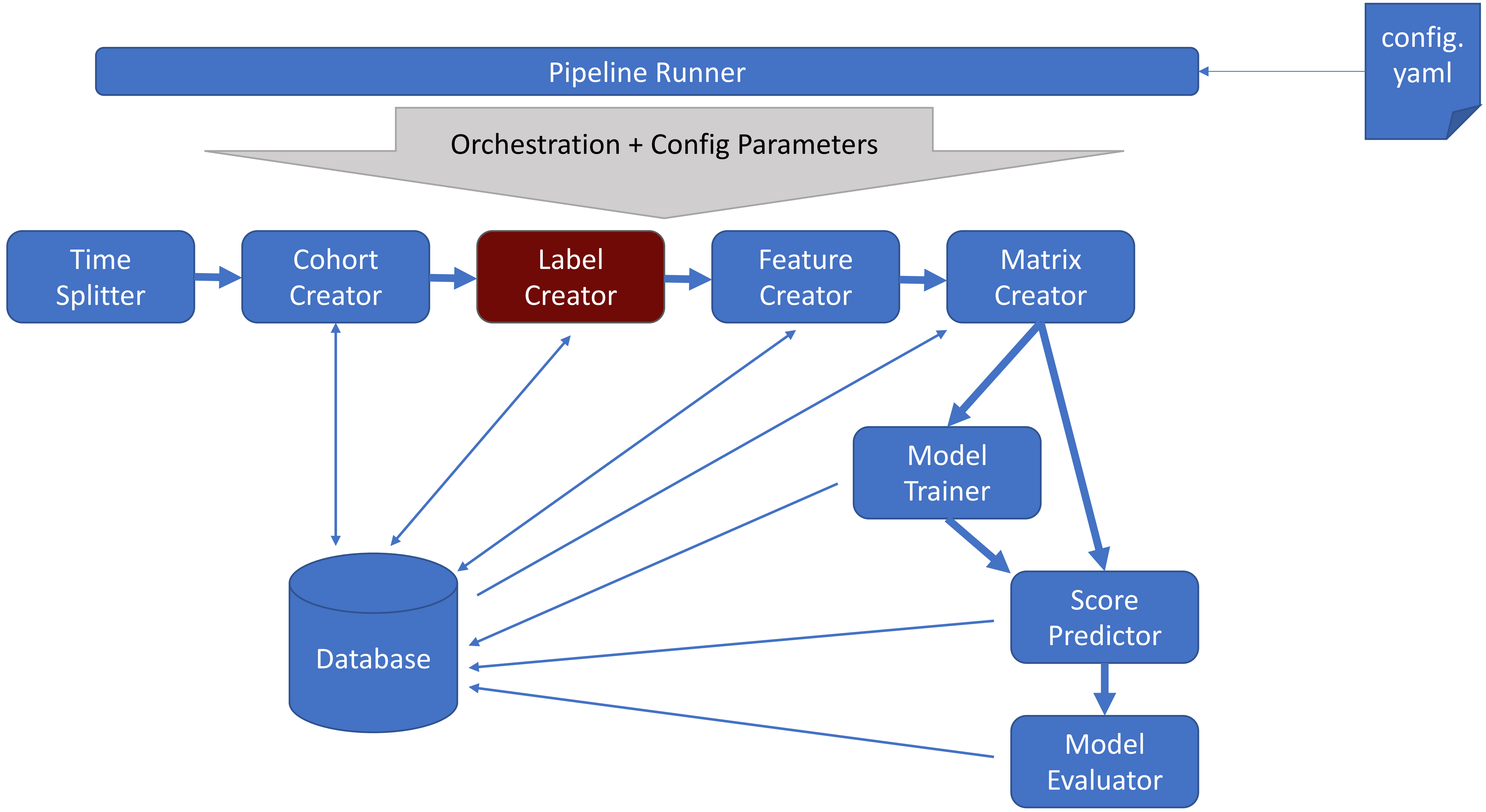
Cohort logic
(e.g., SQL snippet in config)

Cleaned data

OUTPUTS

Sets of entities *at a given time* that will define rows in training/validation matrices:

entity_id, time


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Label Creator

INPUTS

Train/validate split dates

Cohort ids+dates

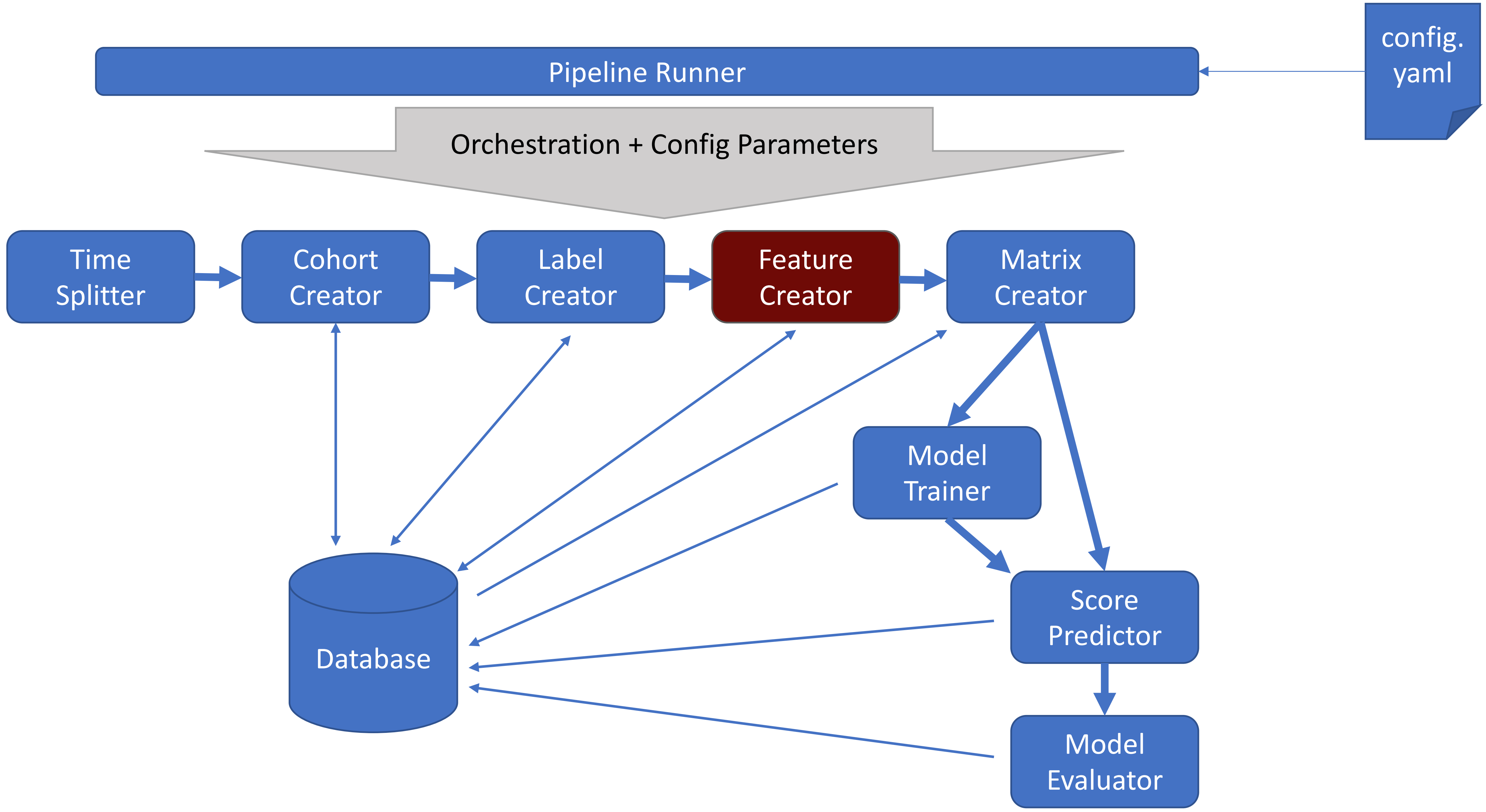
Label definition logic + window
(e.g., via SQL snippet in config)

Cleaned data

OUTPUTS

Label values for each entity/date pair in the cohort:

entity_id, date, label(s)


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Feature Creator

INPUTS

Train/validate split dates

Cohort ids+dates

Feature definition logic + windows
(e.g., via SQL snippets in config)

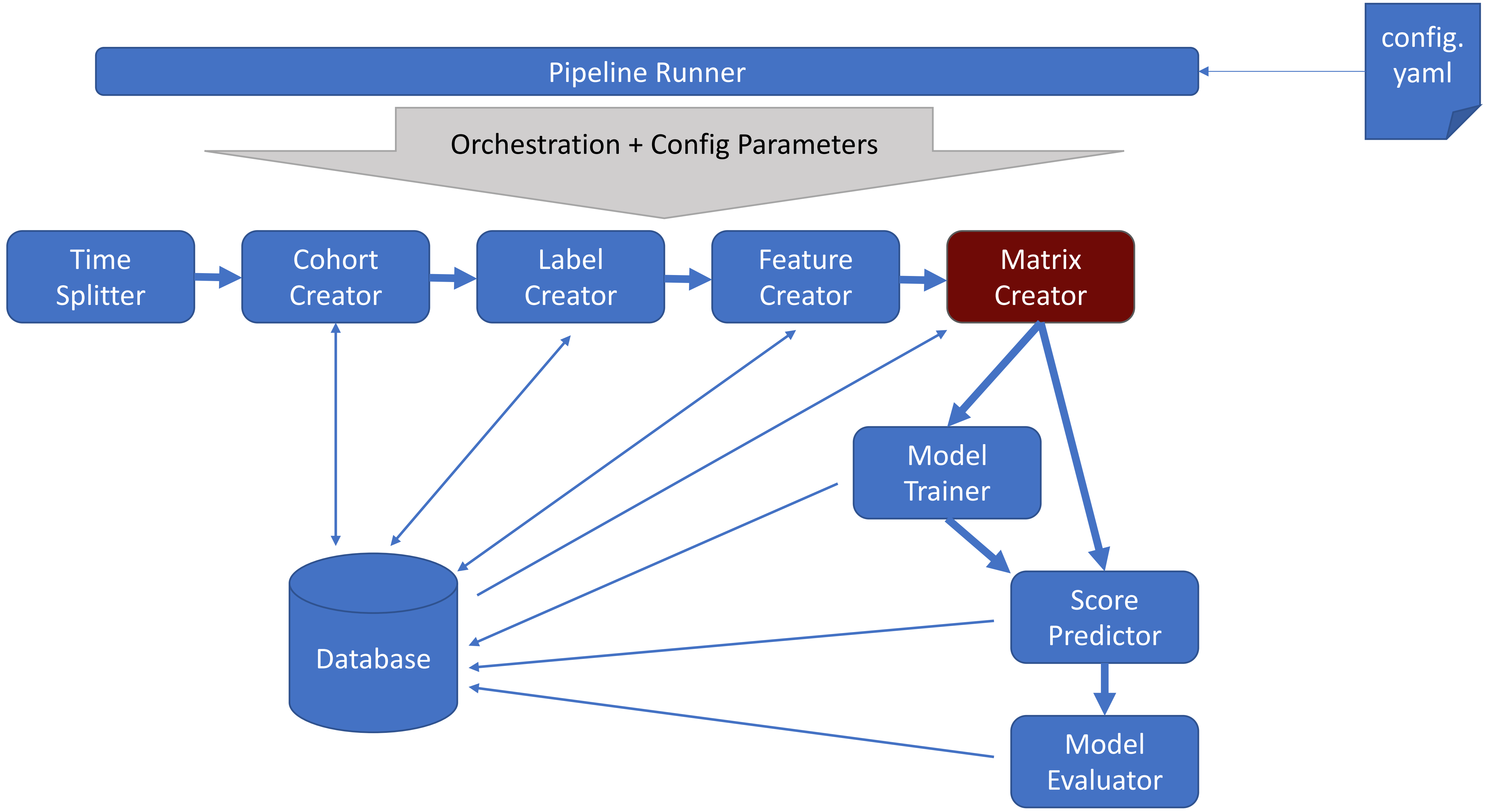
Cleaned data

OUTPUTS

Feature values for each entity/date pair in the cohort:

entity_id, date, feature cols

Note: often useful to group related features together for testing, etc.


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Matrix Creator

INPUTS

Train/validate split dates

Cohort ids+dates

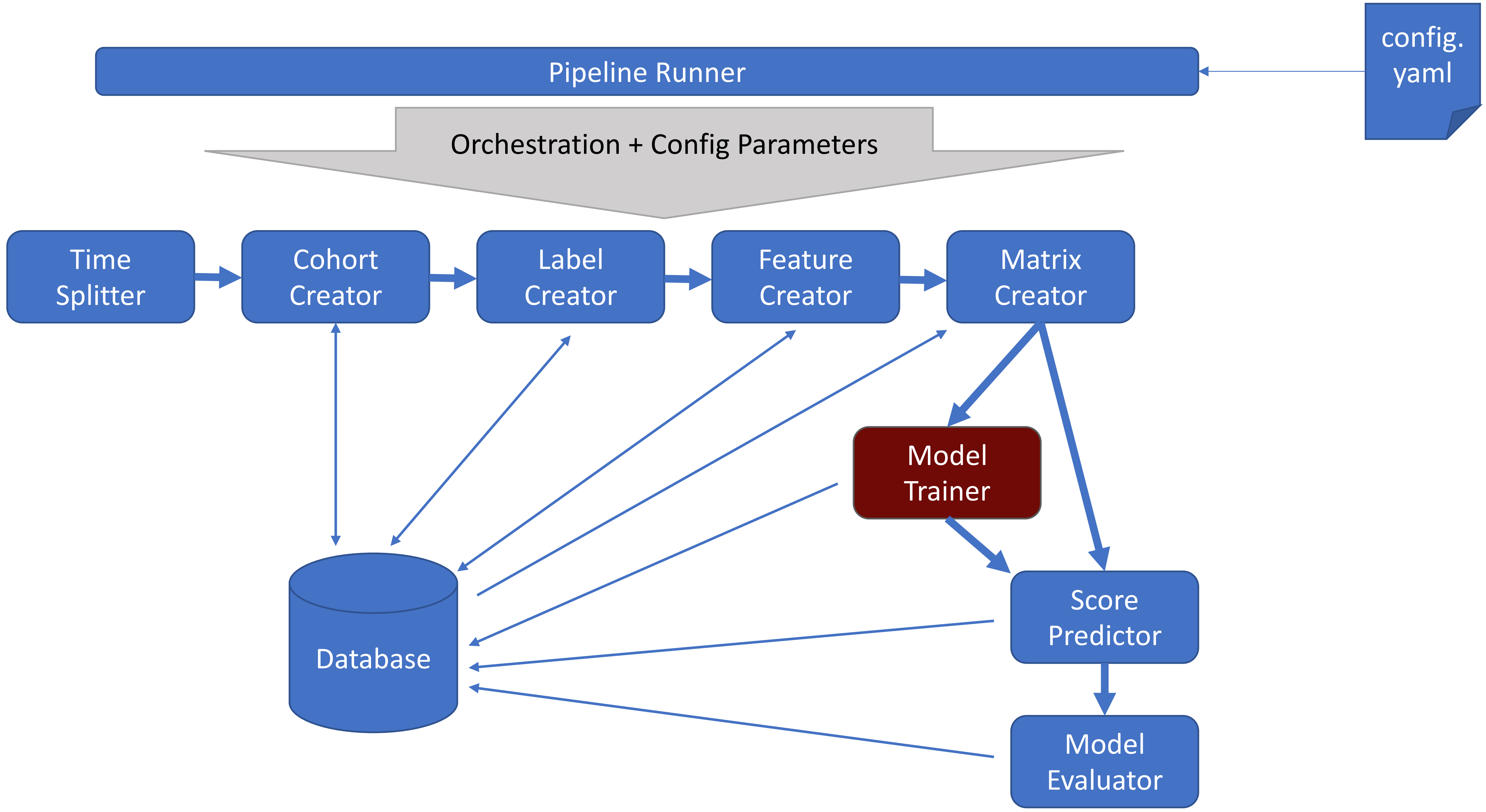
Label values

Feature values

OUTPUTS

Pairs of train + validation matrices
(np.array, pd.DataFrame, scipy.csr_matrix, etc.)

Note: all this needs to do at this point is join the cohorts, labels, and features together for each set of dates


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Model Trainer

INPUTS

Train Matrix
(and associated temporal metadata)

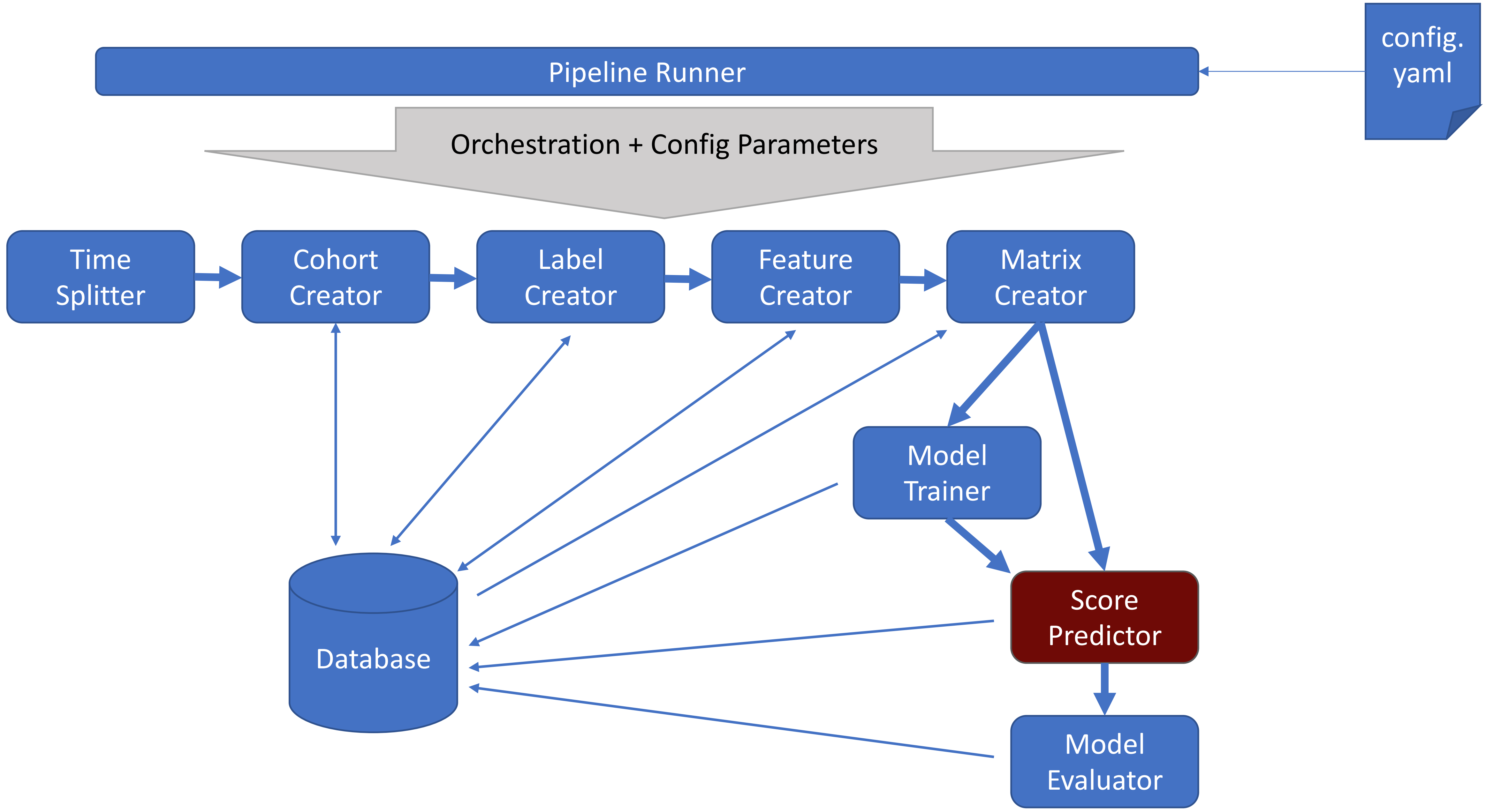
Model type + parameters
(from config)

OUTPUTS

Trained model object
(downstream + to disk)

Model metadata to database
model type, hyperparameters,
training dates, model_id

Note: Helpful to keep track of sets of models with the same parameters (type, features, hyperparameters, label definition, etc) trained on different time splits


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Score Predictor

INPUTS

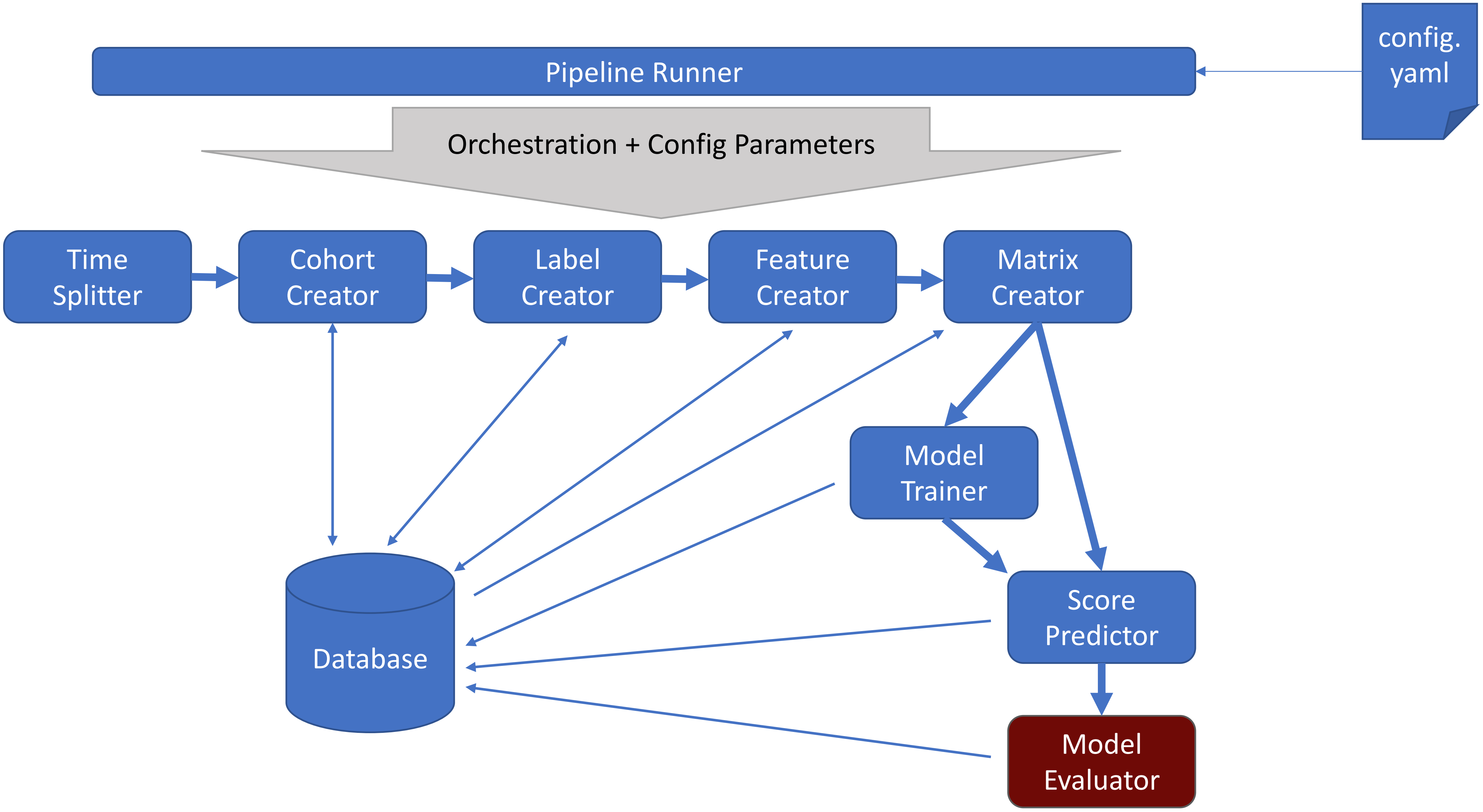
Validation Matrix
(and associated temporal metadata)

Trained model object
(and associated model_id)

OUTPUTS

Predicted scores

Note: Helpful to persist to the database for downstream analyses, but can get large, so be sure to index (and may want to make storage optional, especially during initial debugging)



Pipeline Runner
config.
yaml
Orchestration + Config Parameters
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

Model Evaluator

INPUTS

Validation Matrix (Actual Labels)
(and associated temporal metadata)

Predicted Scores

Metric definition(s)
(from config)

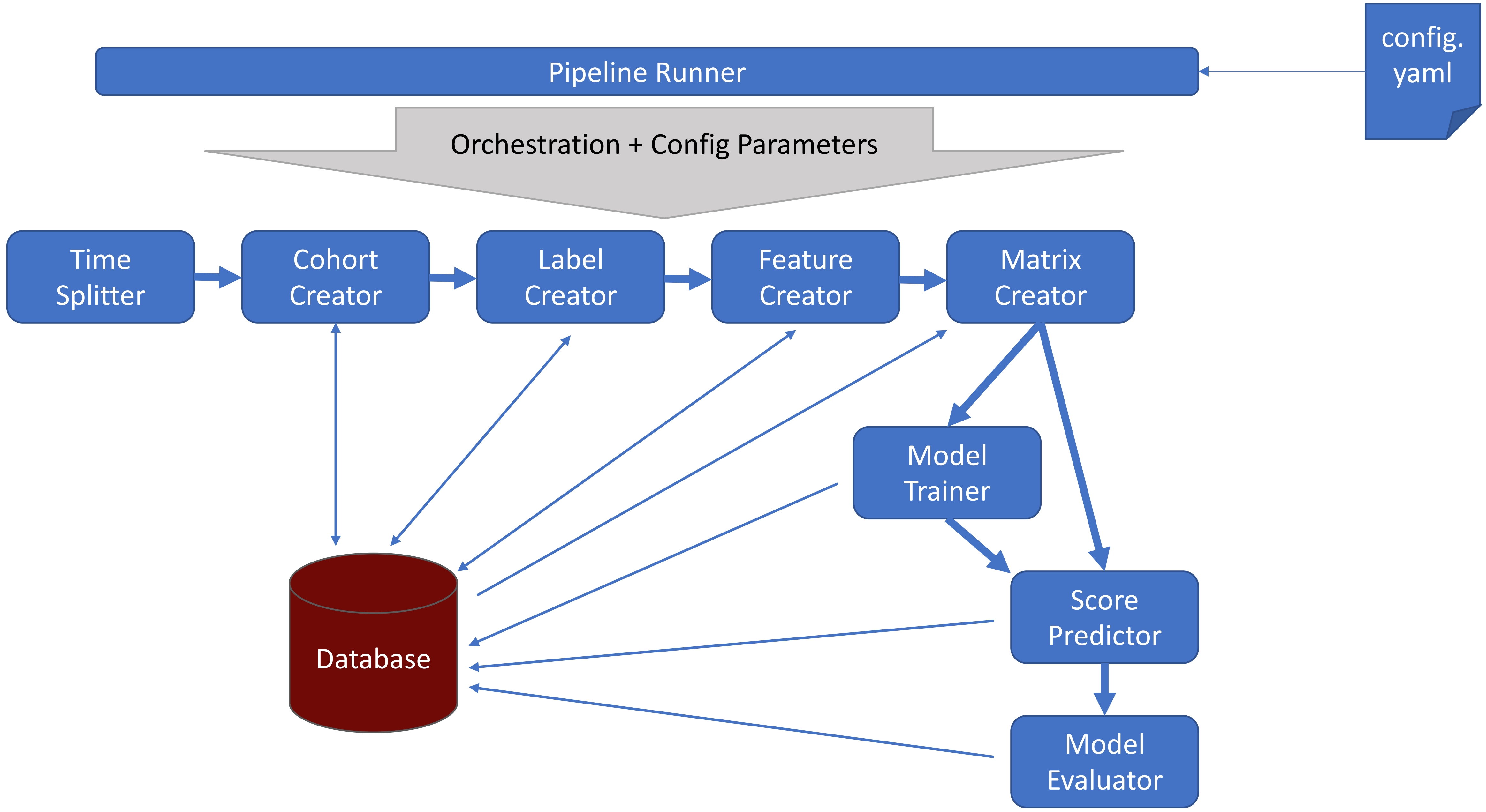
OUTPUTS

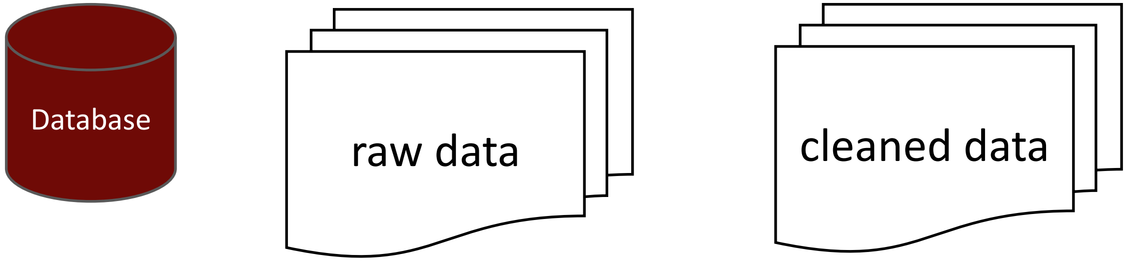
Model performance on metrics:
model_id, validation_date, metric, parameter, value

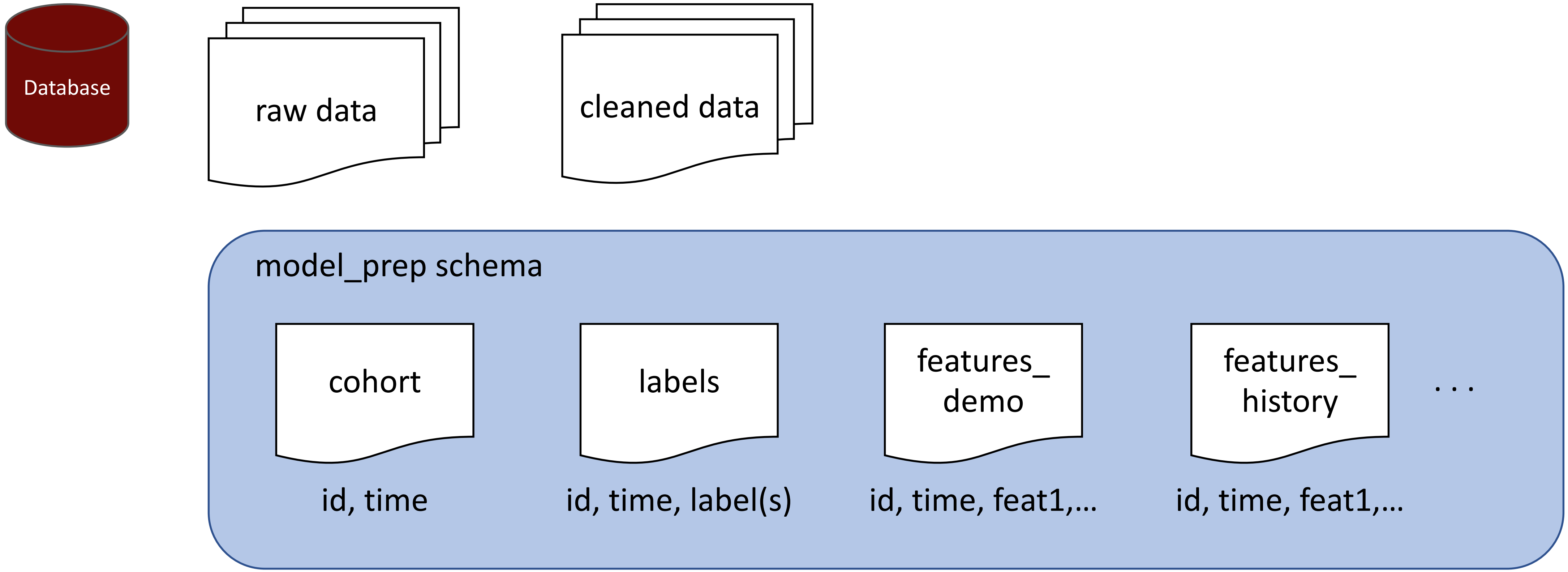
examples:

108, 2015-03-14, precision, 500_abs, 0.62
108, 2015-03-14, recall, 15_pct, 0.25
108, 2015-03-14, recall, 0.8_thresh, 0.42

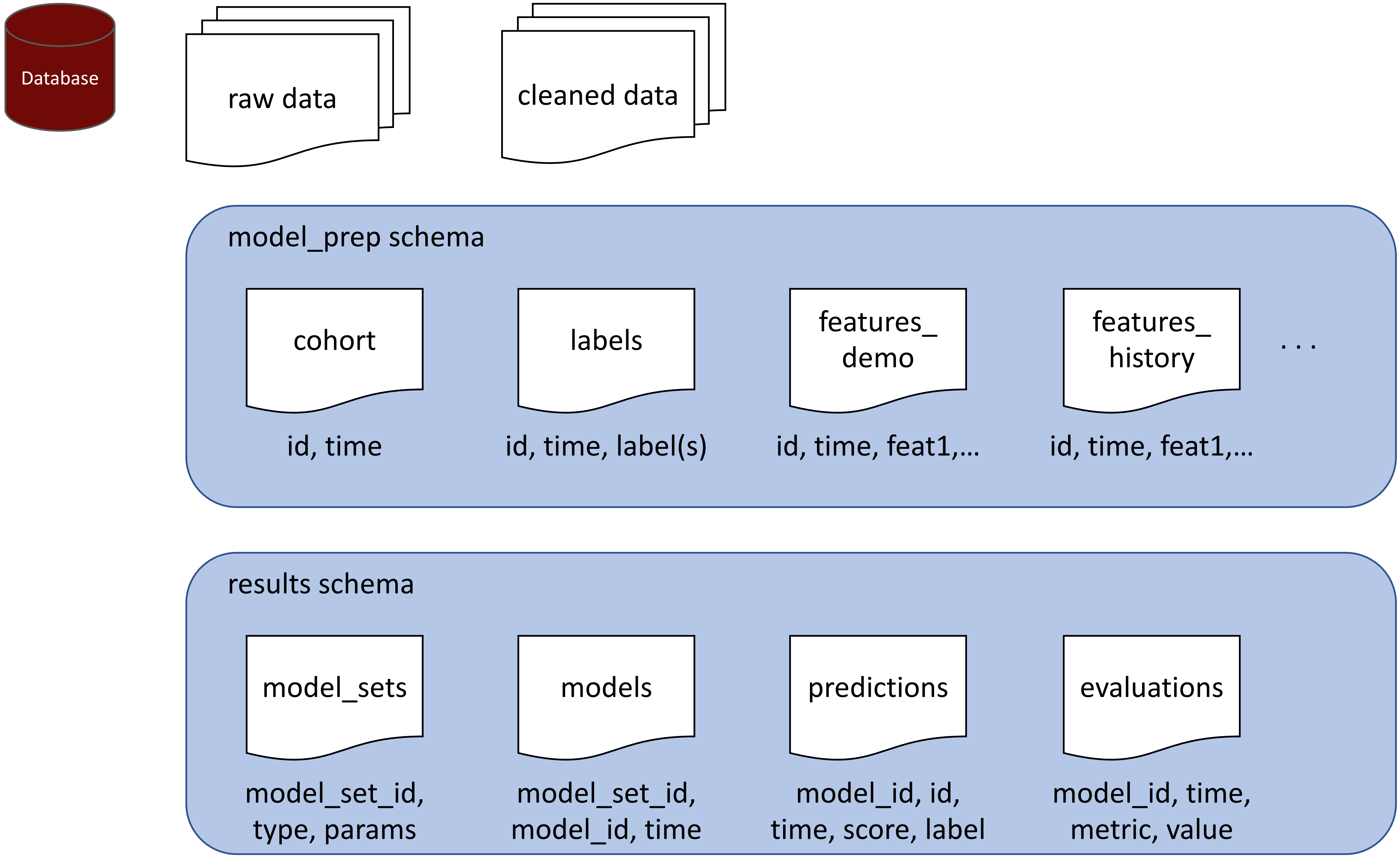
Note: Helpful to persist to the database for downstream analyses

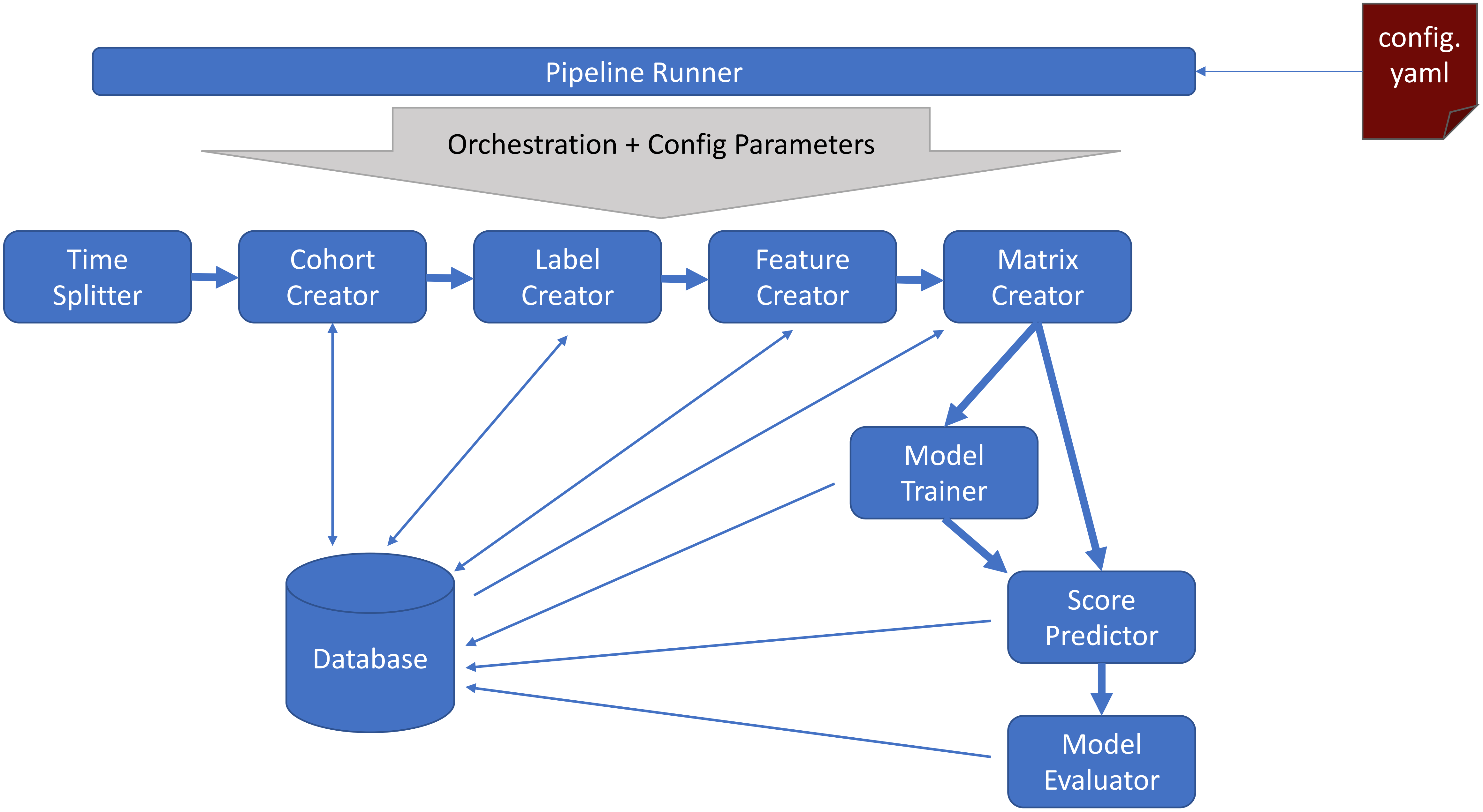

config. yaml
Pipeline Runner
Orchestration + Config Parameters
Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Score Predictor
Model Evaluator
Database

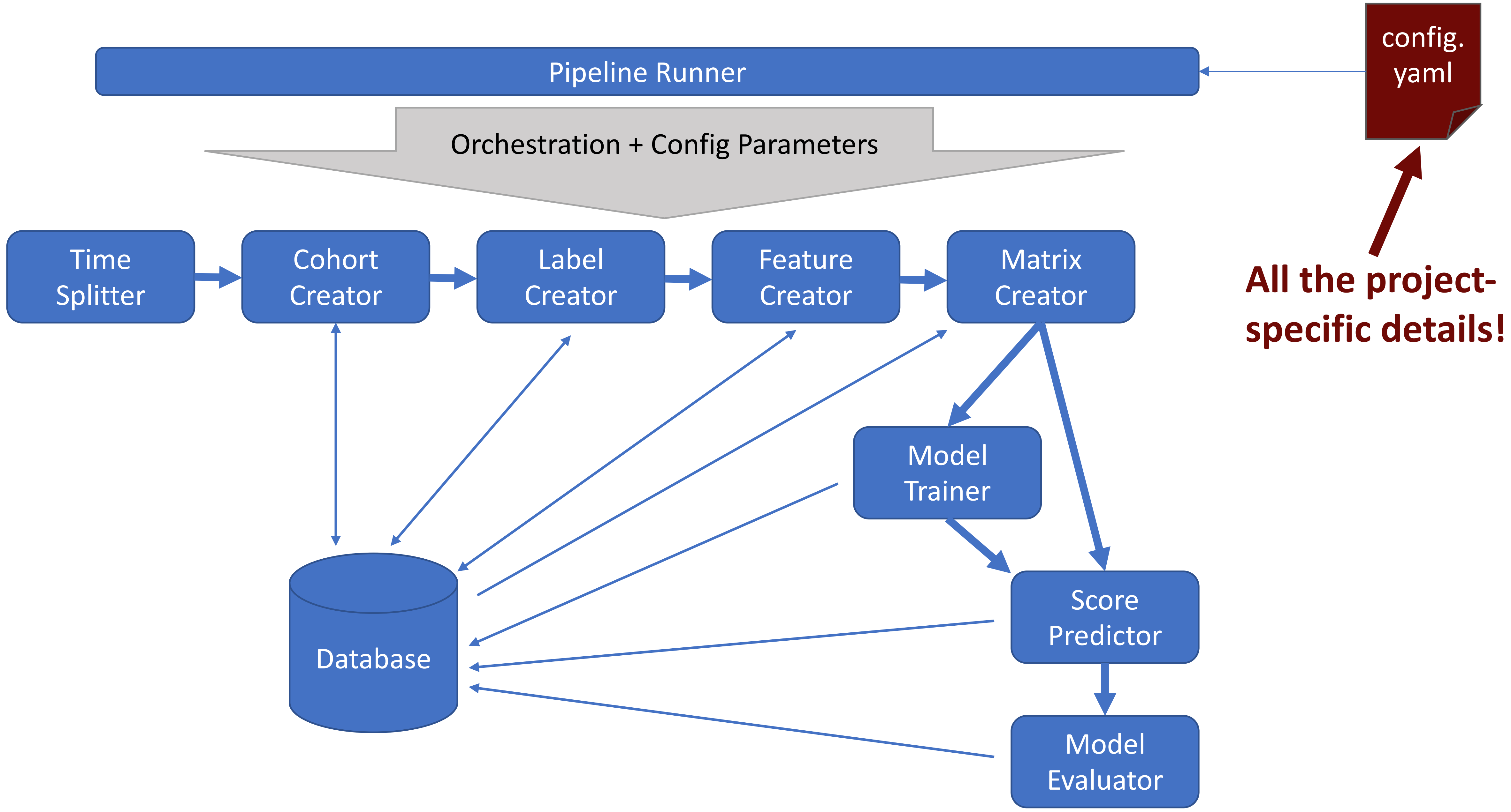

Database
raw data
cleaned data



Database
raw data
cleaned data
model_prep schema
cohort
labels
features_ demo
features_ history
. . .
id, time
id, time, label(s)
id, time, feat1,…
id, time, feat1,…


Database
raw data
cleaned data
model_prep schema
cohort
id, time
labels
id, time, label(s)
features_ demo
id, time, feat1,…
features_ history
id, time, feat1,…
. . .
results schema
model_sets
model_set_id, type, params
models
model_set_id, model_id, time
predictions
model_id, id, time, score, label
evaluations
model_id, time, metric, value


config. yaml
Pipeline Runner
Orchestration + Config Parameters
Time Splitter
Cohort Creator
Label Creator
Feature Creator
Matrix Creator
Model Trainer
Database
Score Predictor
Model Evaluator


config.
yaml
Pipeline Runner
Orchestration + Config Parameters
All the project-
specific details!
Time
Splitter
Cohort
Creator
Label
Creator
Feature
Creator
Matrix
Creator
Model
Trainer
Score
Predictor
Model
Evaluator
Database

config. yaml


```yaml
config_version: 'v6'

model_comment: 'company_inspected (test)'

temporal_config:
    feature_start_time: '2015-01-01' # earliest date included in features
    feature_end_time: '2018-01-01'   # latest date included in features
    label_start_time: '2015-01-01' # earliest date for which labels are avialable
    label_end_time: '2018-01-01' # day AFTER last label date (all dates in any model are < this date)
    model_update_frequency: '1y' # how frequently to retrain models
    training_as_of_date_frequencies: '1y' # time between as of dates for same entity in train matrix
    test_as_of_date_frequencies: '1month' # time between as of dates for same entity in test matrix
    max_training_histories: ['10y'] # length of time included in a train matrix
    test_durations: ['0day'] # length of time included in a test matrix (0 days will give a single prediction immediately after training end)
    training_label_timespans: ['1month'] # time period across which outcomes are labeled in train matrices
    test_label_timespans: ['1month'] # time period across which outcomes are labeled in test matrices

cohort_config:
    query: |
      select distinct on (company)
      company as entity_id
      from semantic.companies
      where end_date is null or end_date <= '{as_of_date}'::date
    name: 'company'

label_config:
    query: |
        select
        company as entity_id,
        1::smallint as outcome
        from semantic.events
        where '{as_of_date}'::date <= inspection_start_date
        and inspection_start_date < '{as_of_date}'::date + interval '{label_timespan}'
        group by company
    include_missing_labels_in_train_as: False
    name: 'inspected'

feature_aggregations:
    -
        prefix: 'companies'
        from_obj: |
          (select company as entity_id, * from semantic.events) as inspections

        knowledge_date_column: 'inspection_start_date'

        aggregates_imputation:
            all:
                type: 'zero'
            max:
                type: 'mean'

        aggregates:
            - # number of events
              quantity:
                total: "*"

              imputation:
                count:
                  type: 'zero'

              coltype: 'smallint'
              metrics: ['count']

        intervals:
            - 'all'
            - '100y'
            - '1y'
            - '1month'
```


https://github.com/dssg/direccion_trabajo_inspections/blob/master/experiments/test.yaml