HTM.core Streamer

Python Module

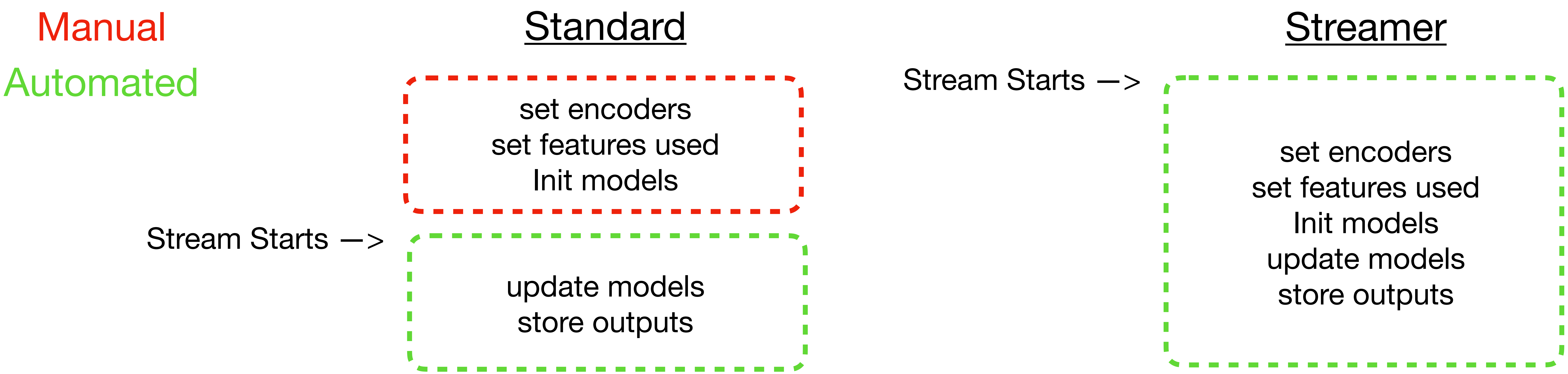
Sam Heiserman — Jan, 2022

Agenda

Questions I'll Answer:	Verification Sought
• What's the module's purpose?	• **Is purpose worthwhile?**
• What logic is used to achieve the purpose?	• **Is logic valid?**
• How is the logic implemented?	• **Is implementation valid?**
• What are known limits & next steps?	• **How to make it more useful/practical?**

Motivation

- Achieve max scalability for **htm.core** thru —> **100% streaming** functionality
  - Eliminate need for EDA by —> generating encoders from stream sample
- Easily control modeling meta-parameters:
  - Whether to —> model each feature separately <u>or</u> all concatenated into 1 model
  - Whether to —> include timestamp feature
  - Whether to —> use predictor
  - Whether to —> disable learning at any point



Manual

Automated

Standard

set encoders
set features used
Init models

Stream Starts —>

update models
store outputs

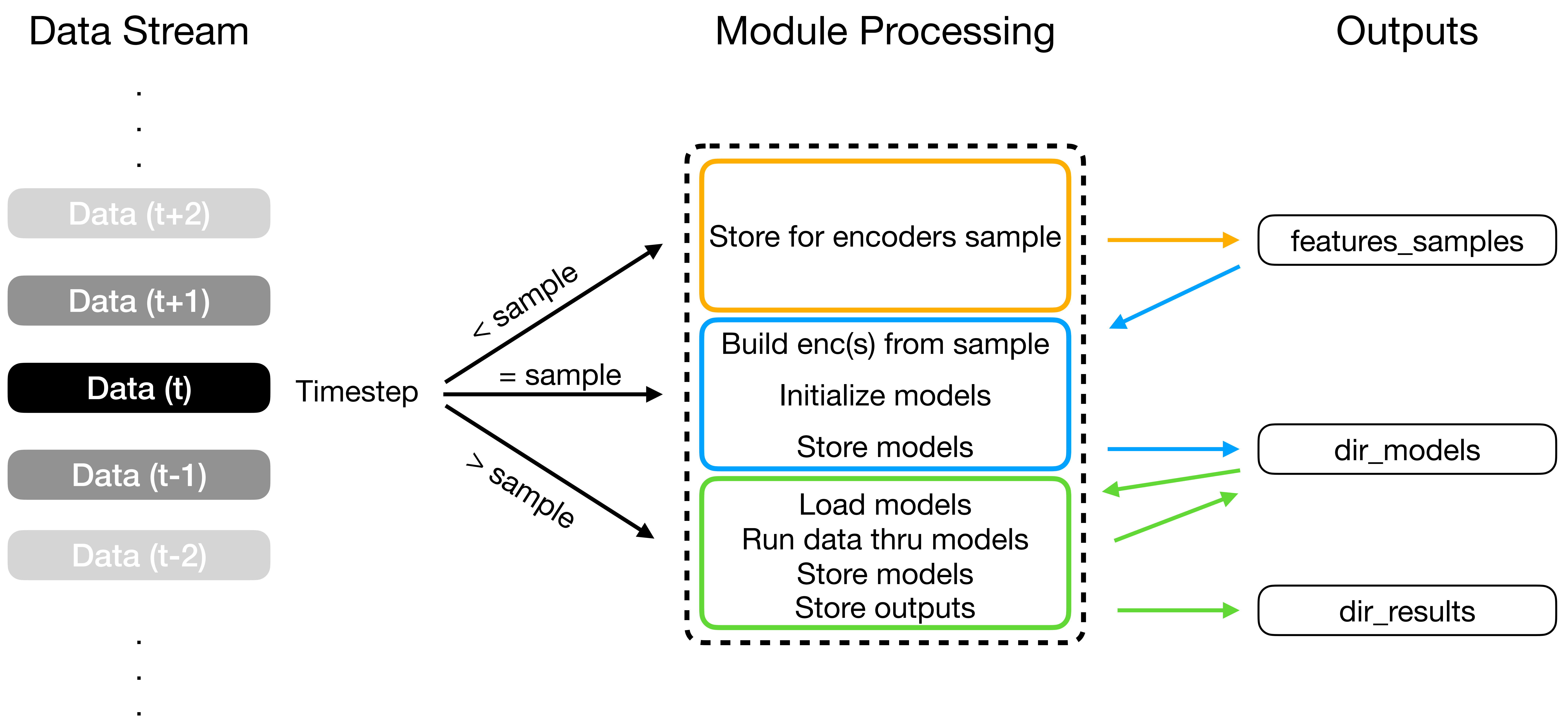
Streamer

Stream Starts —>

set encoders
set features used
Init models
update models
store outputs

Module Overview

1) Sample 2) Initialize 3) Run



Data Stream

Module Processing

Outputs

Pseudocode

Main Functions

htm_stream_runner

```
1. Load —> Config
2. Load —> Batch Data
3. For Row in Batch Data:
    1. Generate —> Stream Data
    2. Store —>  Stream Data
    3. Run —> stream_to_htm()
```

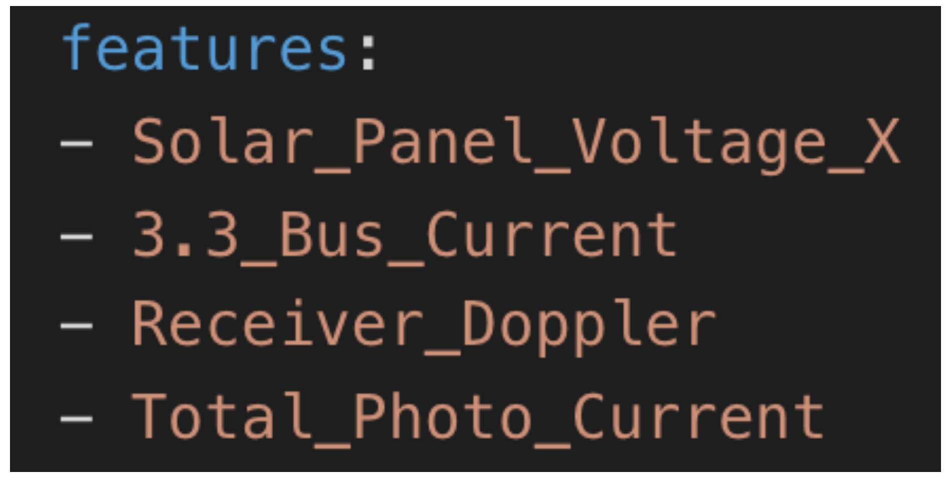
stream_to_htm

```
1. Load —> Config
2. Load —> Data
3. Validate —> Config
if mode == Sample:
    4. Store —> Data
elif mode == Initialize:
    4. Store —> Data
    5. Build —> Encoder Params
    6. Build —> HTM model(s)
    7. Store —> HTM model(s)
else: (mode=Run)
    4. Load —> HTM model(s)
    5. Run —> Data thru HTM model(s)
    6. Store —> HTM outputs
    7. Store —> HTM model(s)
8. Store —> Config
```

Config Structure

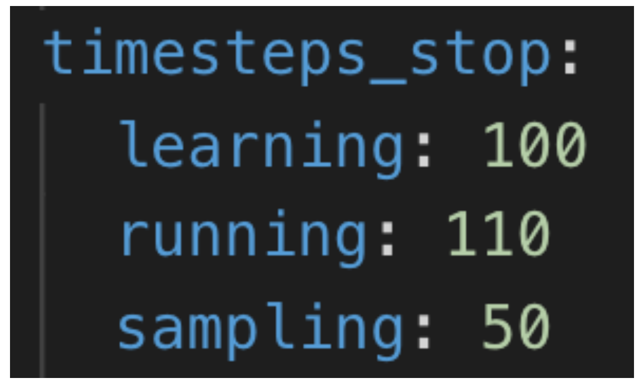
Set by user

Which features are modeled?



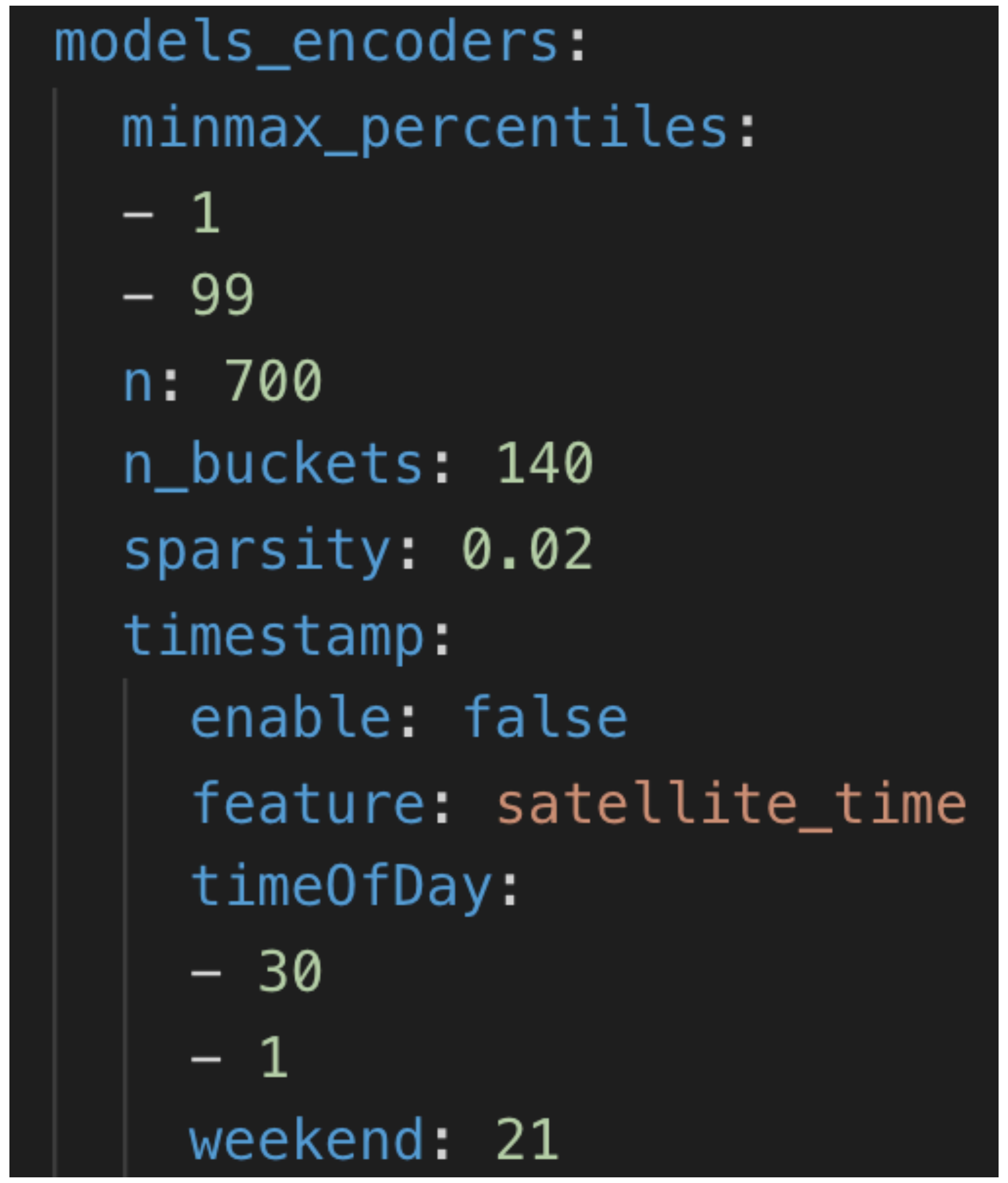
```
features:
- Solar_Panel_Voltage_X
- 3.3_Bus_Current
- Receiver_Doppler
- Total_Photo_Current
```

At what time steps are sampling/learning/running stopped?



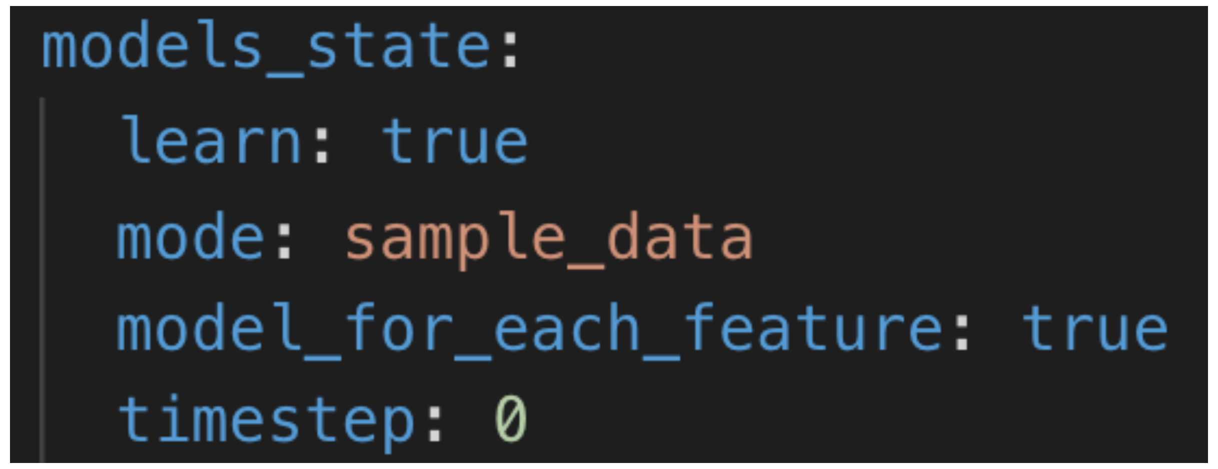
```
timesteps_stop:
  learning: 100
  running: 110
  sampling: 50
```

Are timestamp feature be included in models?
What’s the name of timestamp feature?
What are the encoder params for timestamp?



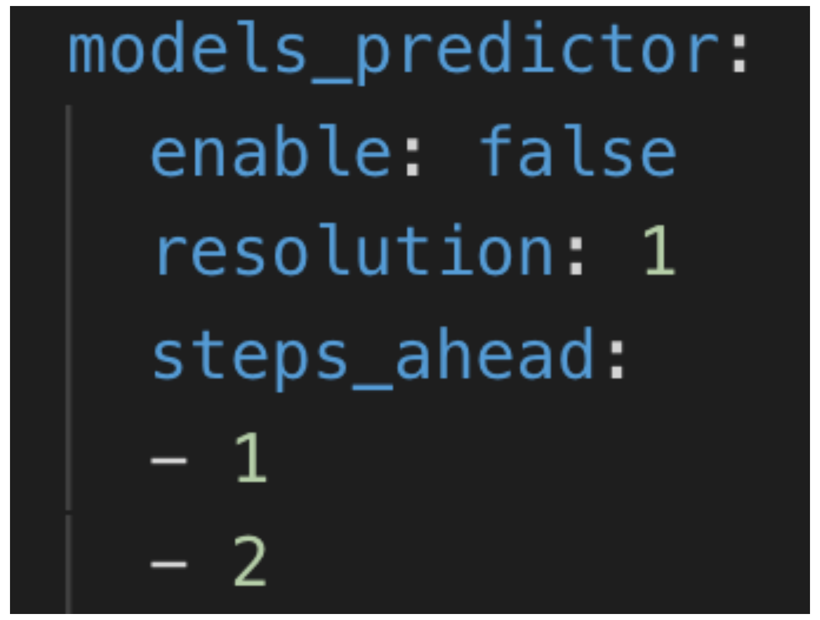
```
models_encoders:
  minmax_percentiles:
  - 1
  - 99
  n: 700
  n_buckets: 140
  sparsity: 0.02
  timestamp:
    enable: false
    feature: satellite_time
    timeOfDay:
    - 30
    - 1
    weekend: 21
```

Is there a model for each feature, or one model combing all?



```
models_state:
  learn: true
  mode: sample_data
  model_for_each_feature: true
  timestep: 0
```

Is the htm.core predictor be active?
What is predictor resolution?
How many steps ahead does predictor go?



```
models_predictor:
  enable: false
  resolution: 1
  steps_ahead:
  - 1
  - 2
```

Config Structure

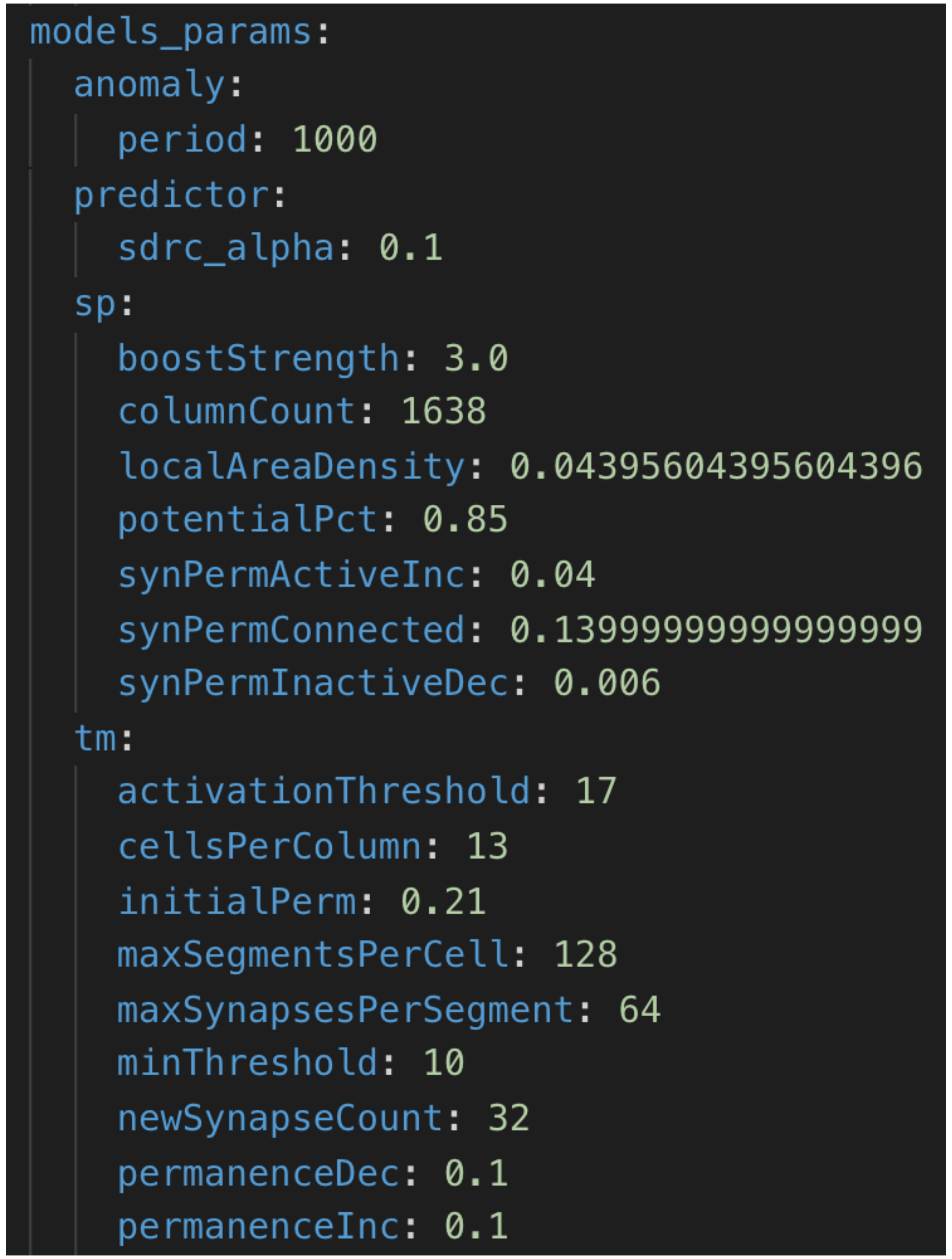
Set by user

What are the params for htm.core.AnomalyLikelihood?

What are the params for htm.core.Predictor?

What are the params for htm.core.SpatialPooler?

What are the params for htm.core.TemporalMemory?



```
models_params:
  anomaly:
    period: 1000
  predictor:
    sdrc_alpha: 0.1
  sp:
    boostStrength: 3.0
    columnCount: 1638
    localAreaDensity: 0.04395604395604396
    potentialPct: 0.85
    synPermActiveInc: 0.04
    synPermConnected: 0.13999999999999999
    synPermInactiveDec: 0.006
  tm:
    activationThreshold: 17
    cellsPerColumn: 13
    initialPerm: 0.21
    maxSegmentsPerCell: 128
    maxSynapsesPerSegment: 64
    minThreshold: 10
    newSynapseCount: 32
    permanenceDec: 0.1
    permanenceInc: 0.1
```

Limitations

- Encoders
  - Rely on feature distribution stationarity
- Runtime
  - Grows linearly with feature count (assuming 1 model per feature)
  - Slows down a lot when **predictor** active
- Memory
  - Grows a lot when **predictor** active

Next Steps

- Model Monitoring
  - TM connectivity
    - Density of permanent synapses
    - Rate of growth over time
    - Distribution of permanent synapses over columns
  - Feature distributions
    - Drift from original samples — could invalidate encoders
  - Anomaly scores
    - Long periods of 0 or 1.0
  - Prediction counts
    - Long periods of 0 or too high (> 10 ?)

- Quantify performance variation
  - When predictor active
  - When feature counts get big

Function Call Tree

source.pipeline.htm_stream.**stream_to_htm()**

load_config
load_json
validate_config

- if mode == sample:
    - extend_features_samples

- elif mode == initialize:
    - extend_features_samples
    - **build_enc_params**
        - **get_rdse_resolution**
    - **init_models**
        - **HTMModel.init_model()**
            - **HTMModel.init_encs()**
                - htm.core.RDSE_Parameters()
                - htm.core.RDSE()
                - htm.core.DateEncoder()
            - **HTMModel.init_sp()**
                - htm.core.SpatialPooler()
            - **HTMModel.init_tm()**
                - htm.core.TemporalMemory()
            - **HTMModel.init_anomalyhistory()**
                - htm.core.AnomalyLikelihood()
            - **HTMModel.init_predictor()**
                - htm.core.Predictor()
    - save_models
        - save_data_as_pickle

- elif mode == run:
    - load_models
        - load_pickle_object_as_data
    - **run_models**
        - **HTMModel.run()**
            - **HTMMode.get_encoding()**
                - htm.core.encoder.encode()
                - htm.core.SDR.concatenate()
            - htm.core.sp.compute()
            - **HTMModel.get_predcount()**
                - htm.core.tm.activateDendrites()
                - htm.core.tm.getPredictiveCells()
            - htm.core.tm.compute()
            - **HTMModel.get_preds()**
                - htm.core.predictor.infer()
                - htm.core.predictor.learn()
    - save_outputs
    - save_models