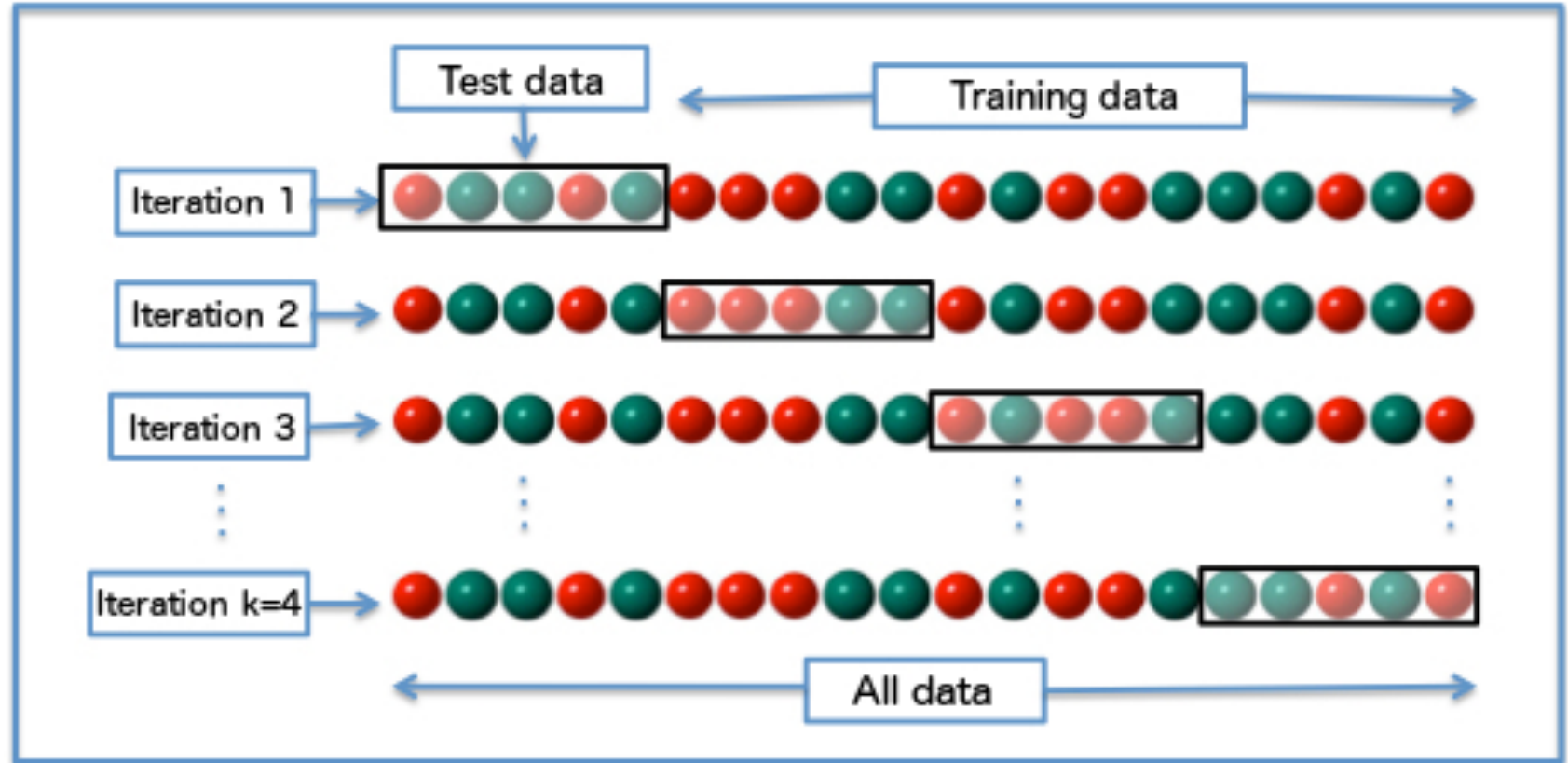
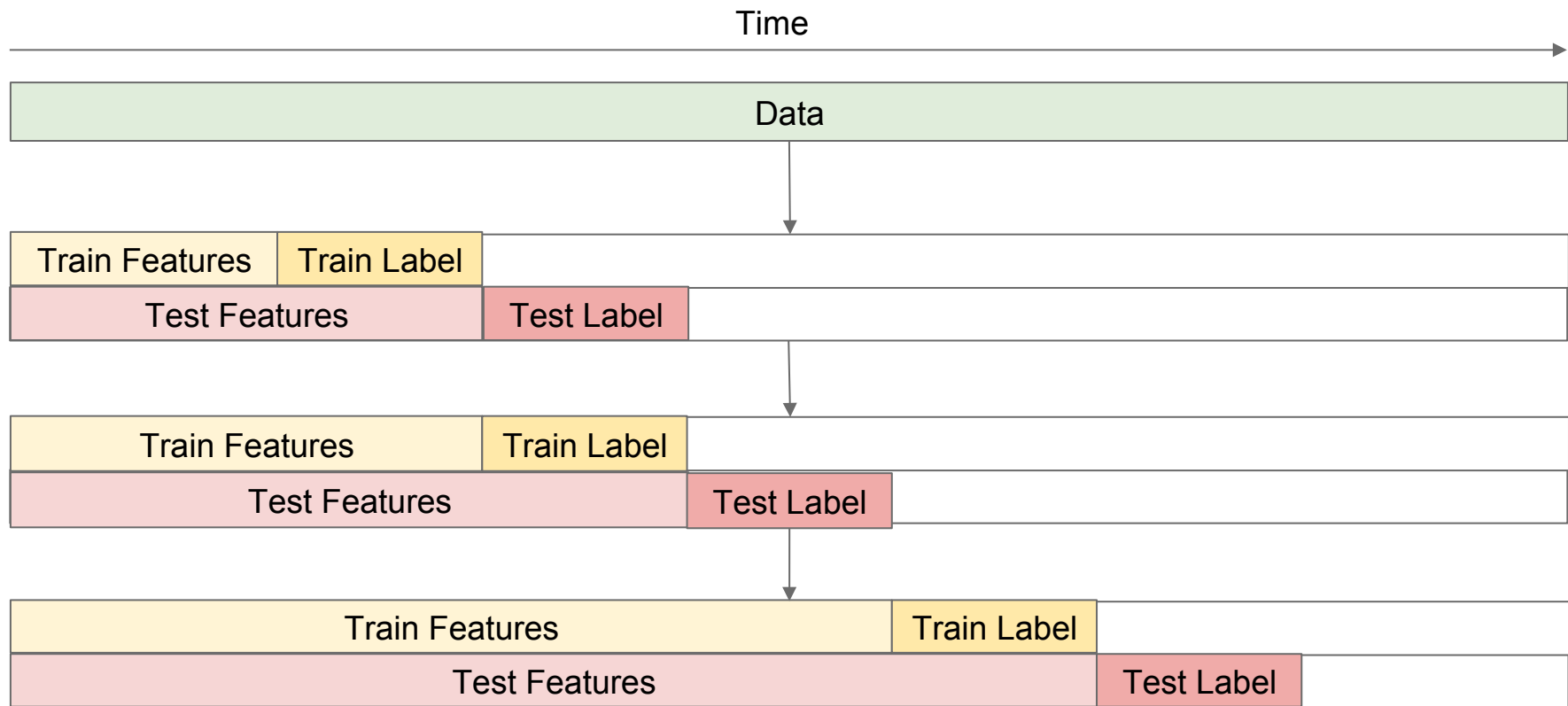


Temporal Cross Validation

Standard Cross Validation



Temporal Cross Validation



Standard

vs

Temporal

- Generally used for non-temporal data, such as for image classification
- Uses comparatively more training data and more folds,
- Theoretically reduces variance in error predictions
- Labels and features are **static properties**

- Especially important for data with a time component
- Mimics the use of our models in the real world
- Computationally cheaper because of reusability of folds
- Theoretically reduces bias in error predictions on out of sample data
- Labels and features are **aggregations over timespans**

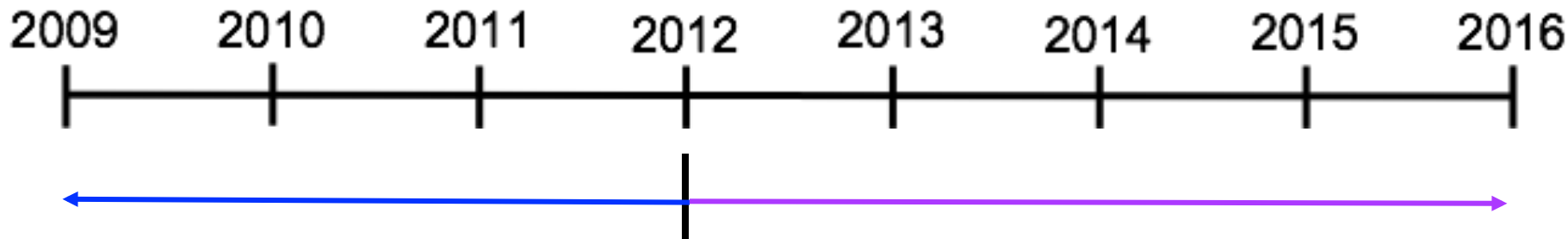
Implementing Temporal Cross Validation

- A series of loops with complicated logic ...

Implementing Temporal Cross Validation

Split data at a specific timestamp so that:

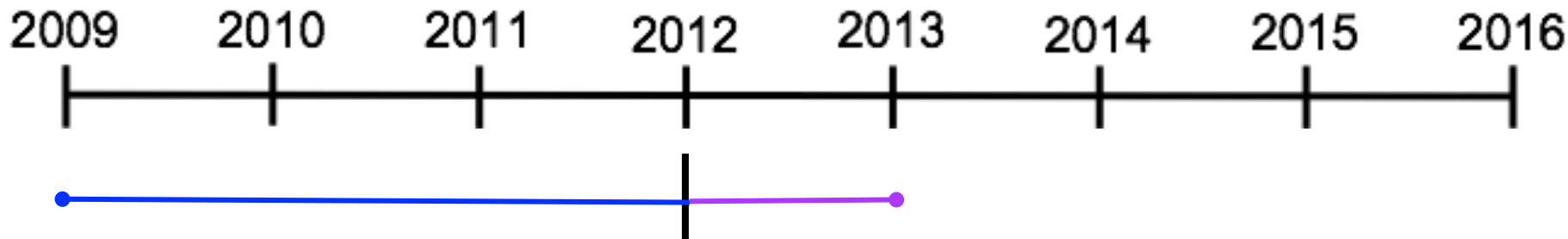
- **features** only incorporate information **before** ($<$) that timestamp
- **labels** include information **at or later** (\geq) than that timestamp



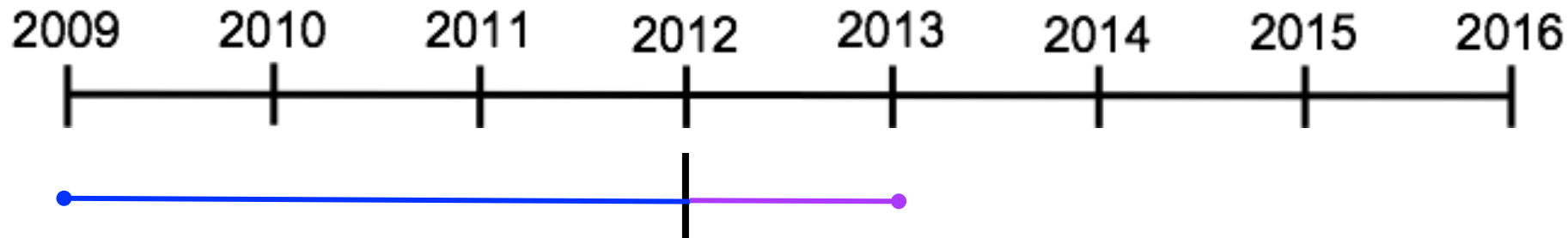
Implementing Temporal Cross Validation

For example:


- **features**: **As of** January 1, 2012, how many jail bookings did you have in the last four years?
- **labels**: **Starting** January 1, 2012, do you have a jail booking in the next year?



entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-01-01	3	1
2	2012-01-01	1	0
3	2012-01-01	1	1




entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-01-01	3	1
2	2012-01-01	1	0
3	2012-01-01	1	1



```
SELECT count(*)  
FROM events  
WHERE event_type = 'booking'  
AND event_date < '2012-01-01'  
AND event_date >= ('2012-01-01 - INTERVAL '4 years')
```

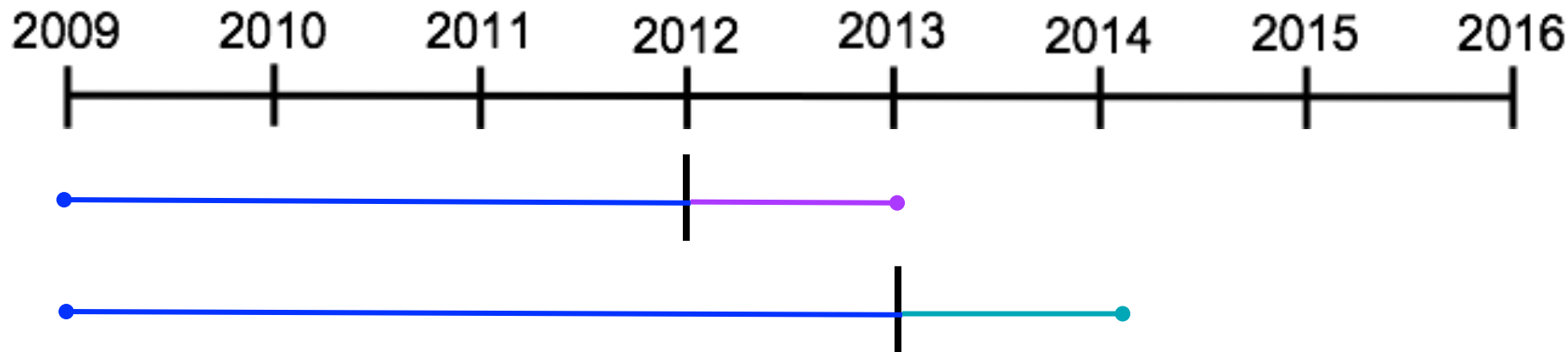
entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-01-01	3	1
2	2012-01-01	1	0
3	2012-01-01	1	1



```
WITH positive_labels AS (  
  SELECT entity_id  
    FROM events  
   WHERE event_type = 'booking'  
         AND event_date >= '2012-01-01'  
         AND event_date < ('2012-01-01 + INTERVAL '1 year')  
)  
SELECT CASE WHEN entity_id IN (positive_labels) THEN 1 ELSE 0 END AS label  
FROM entities
```

Finding Train-Test Splits

Split data at a specific timestamp so that **test labels** do not overlap with **training labels**.

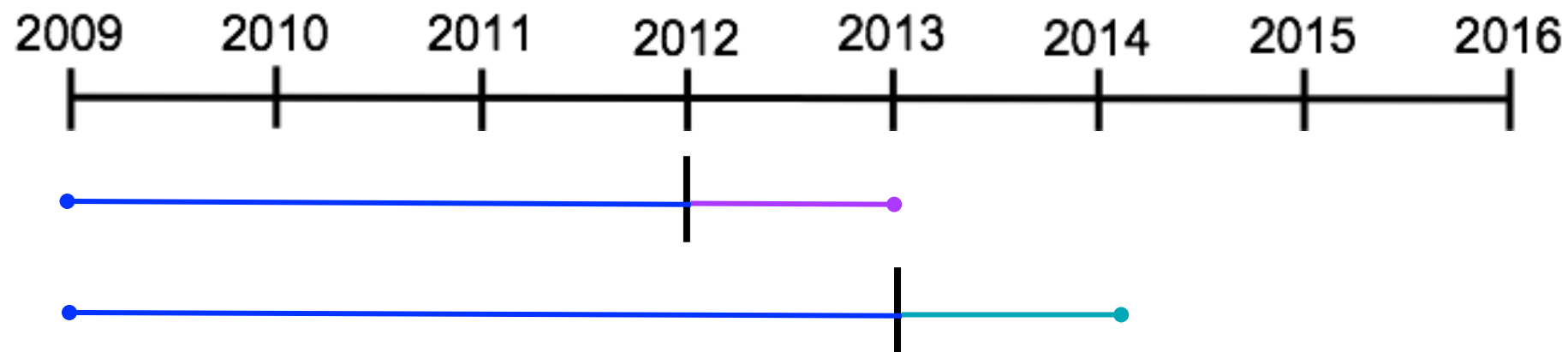


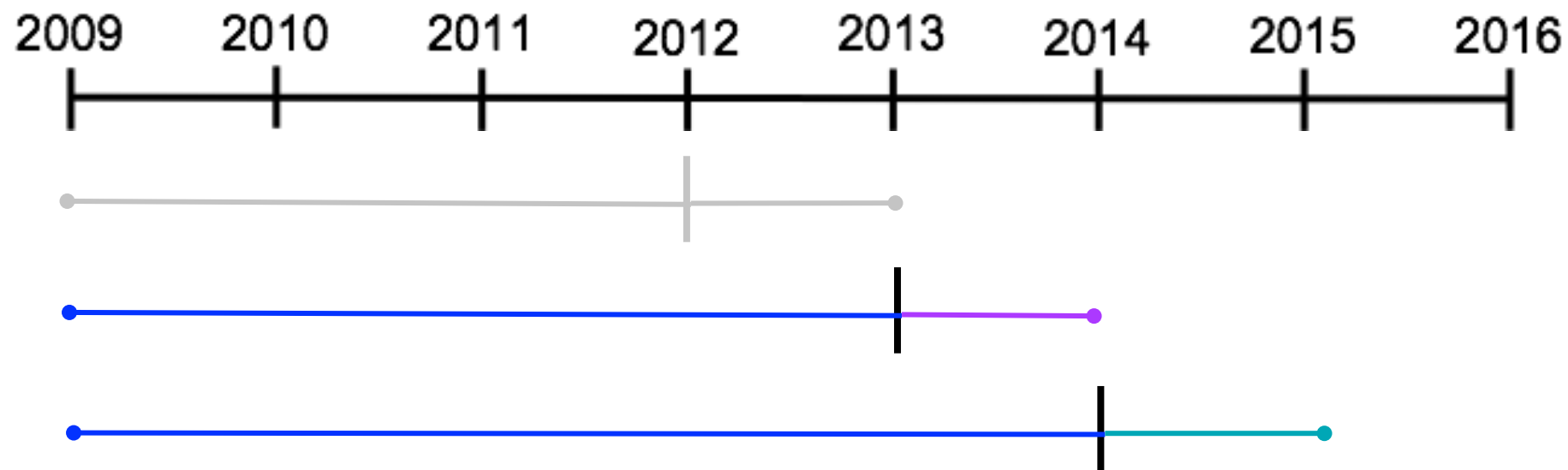
train

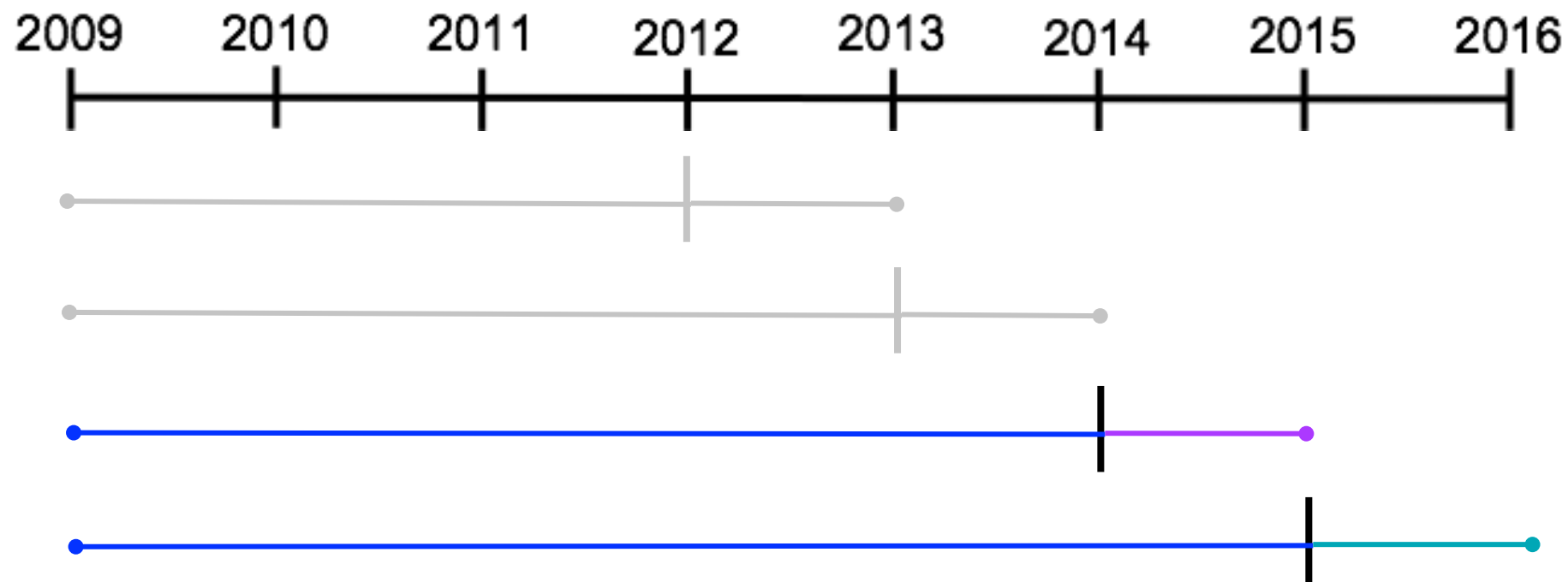
entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-01-01	3	1
2	2012-01-01	1	0
3	2012-01-01	1	1

test

entity_id	as_of_date	bookings_last_4_yrs	label
1	2013-01-01	2	1
2	2013-01-01	2	1
3	2013-01-01	0	0



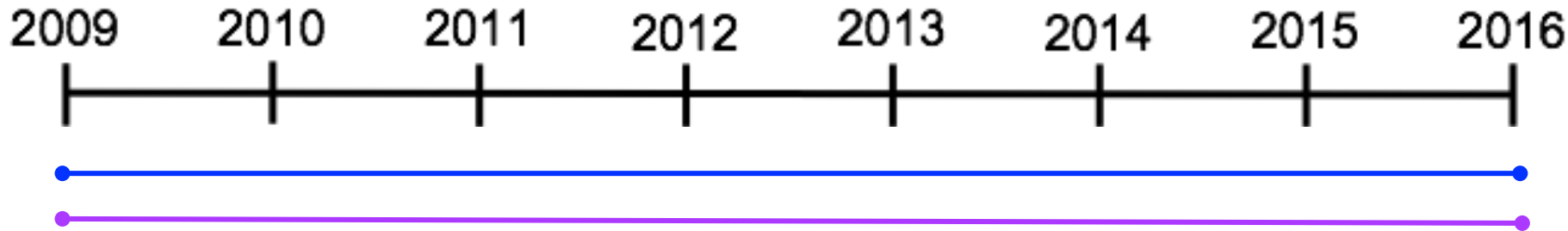




Train-Test Splits: How Much Time for Modeling?

Start and end times:

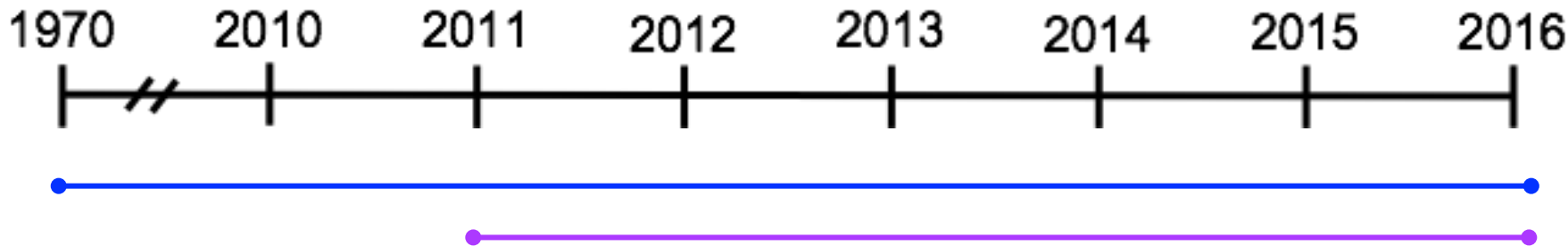
- **feature_start_time**: earliest time in any feature aggregation
- **feature_end_time**: upper limit for data in any feature aggregation
- **label_start_time**: earliest time in any label
- **label_end_time**: upper limit for label data

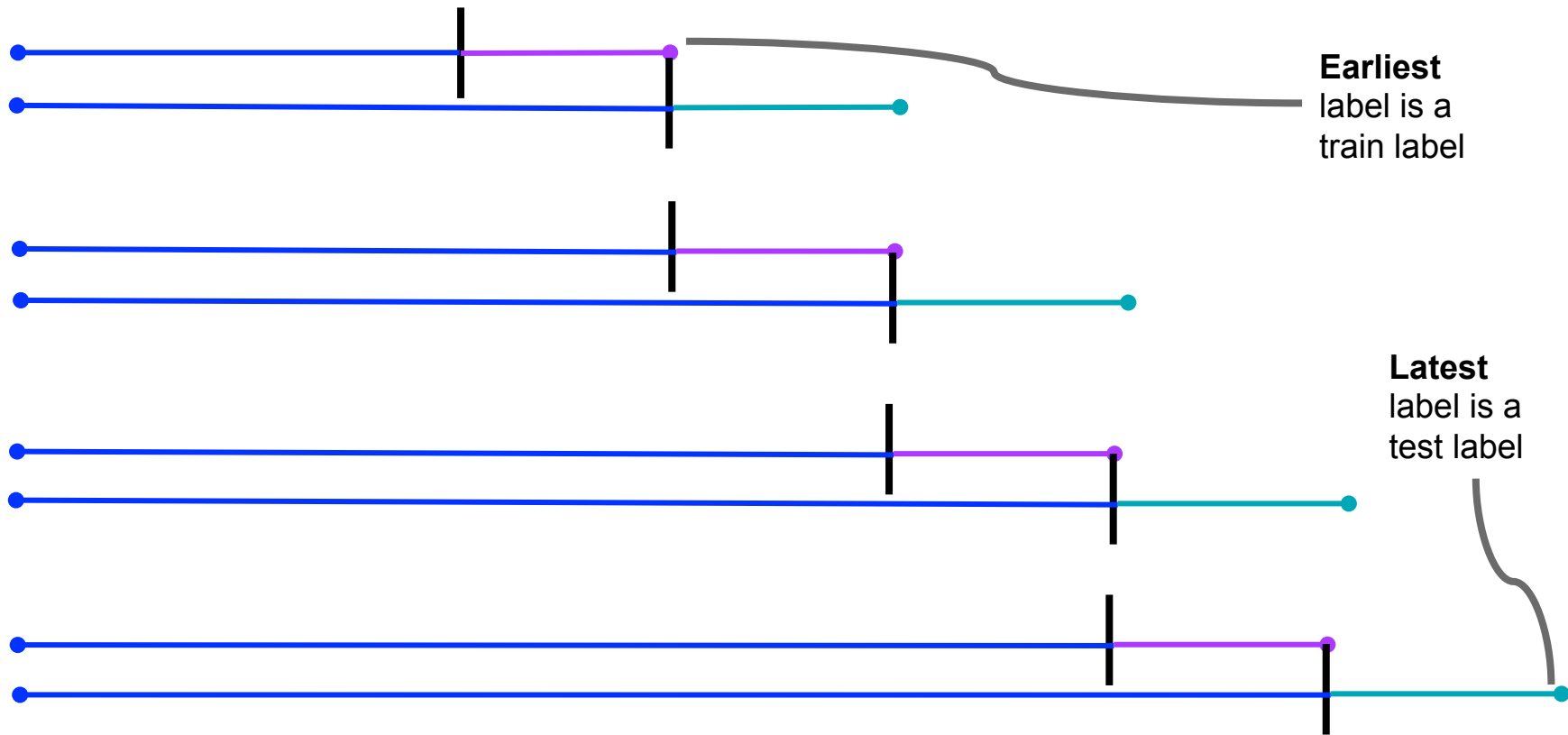
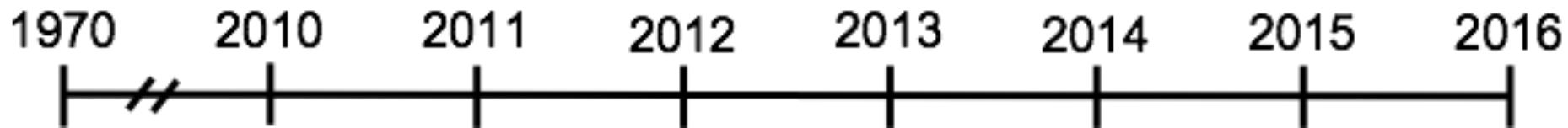


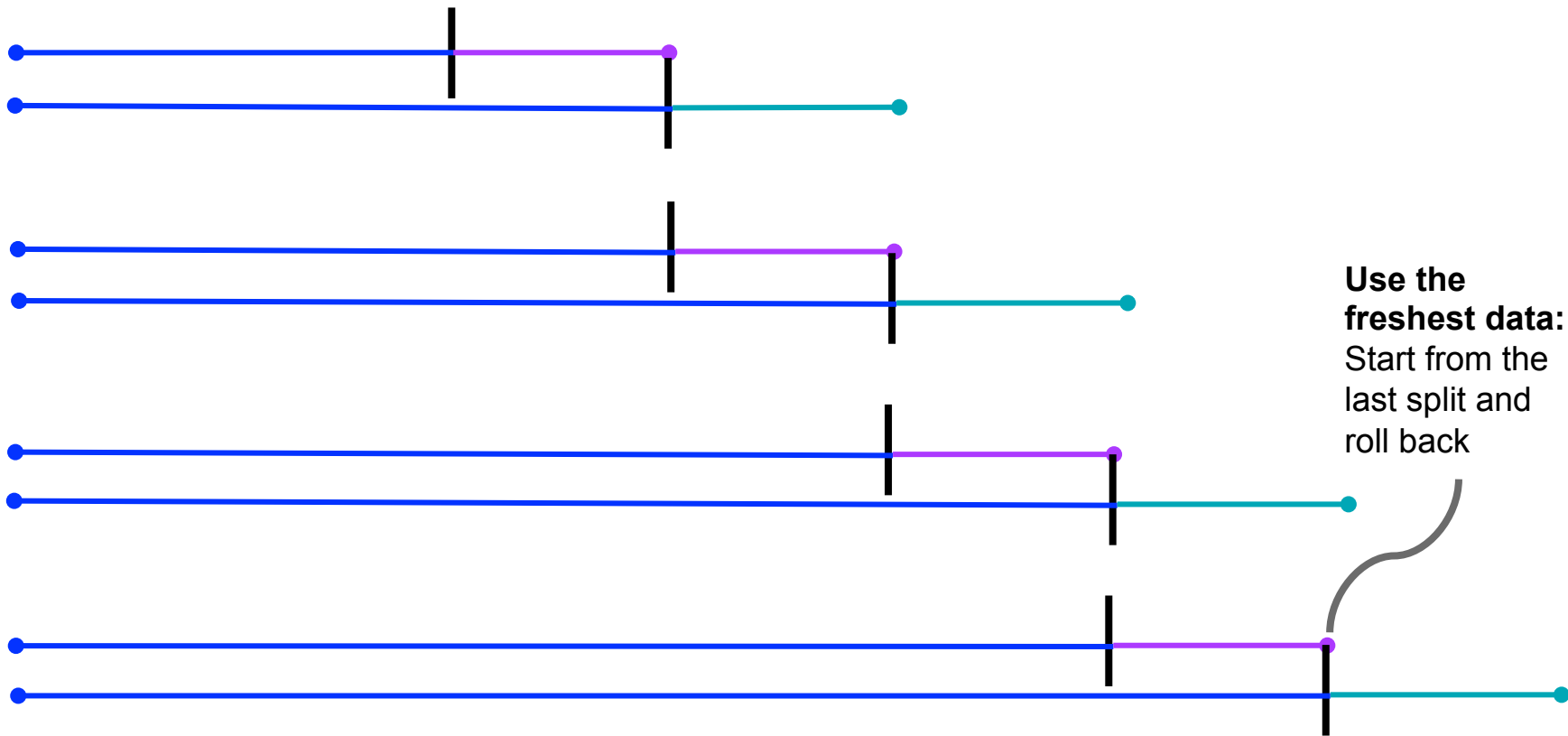
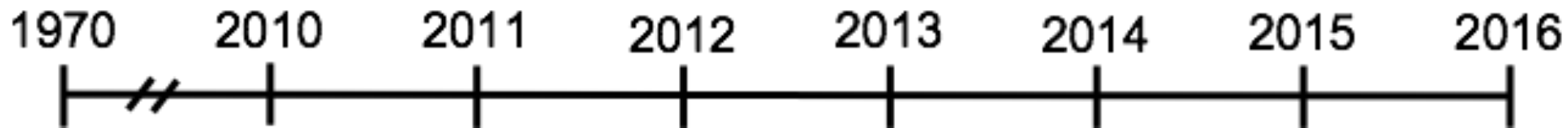
Train-Test Splits: How Much Time for Modeling?

For example:

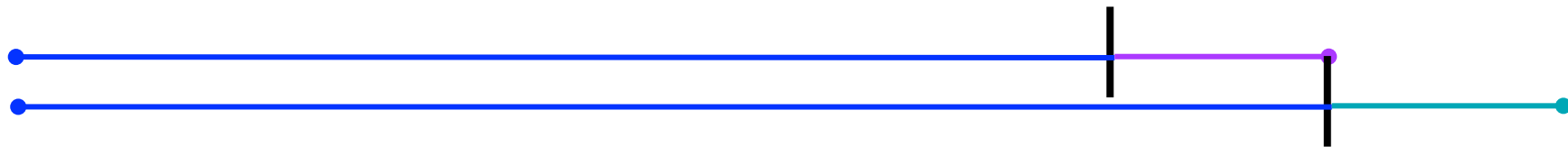
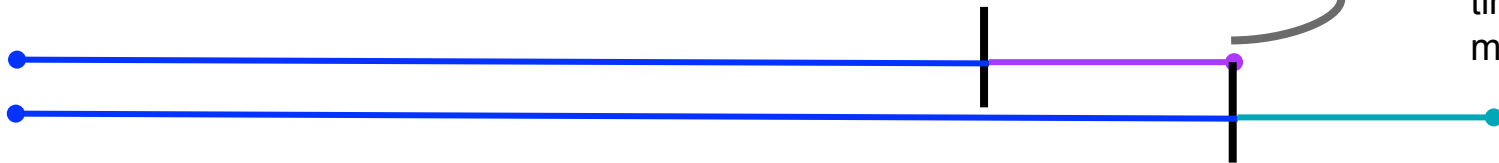
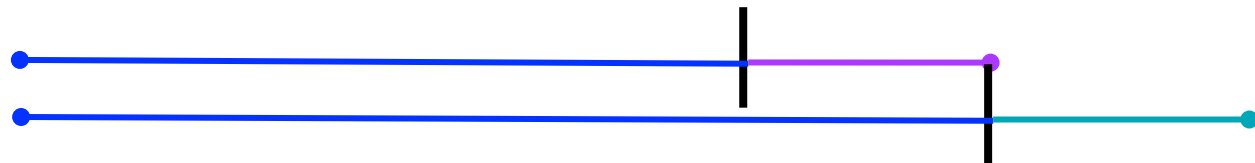
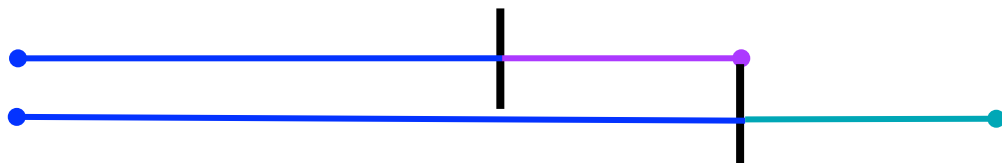
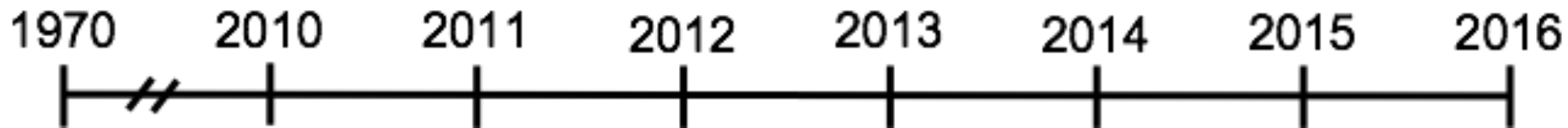
- **feature_start_time**: January 1, 1970
- **feature_end_time**: January 1, 2016
- **label_start_time**: January 1, 2011
- **label_end_time**: January 1, 2016



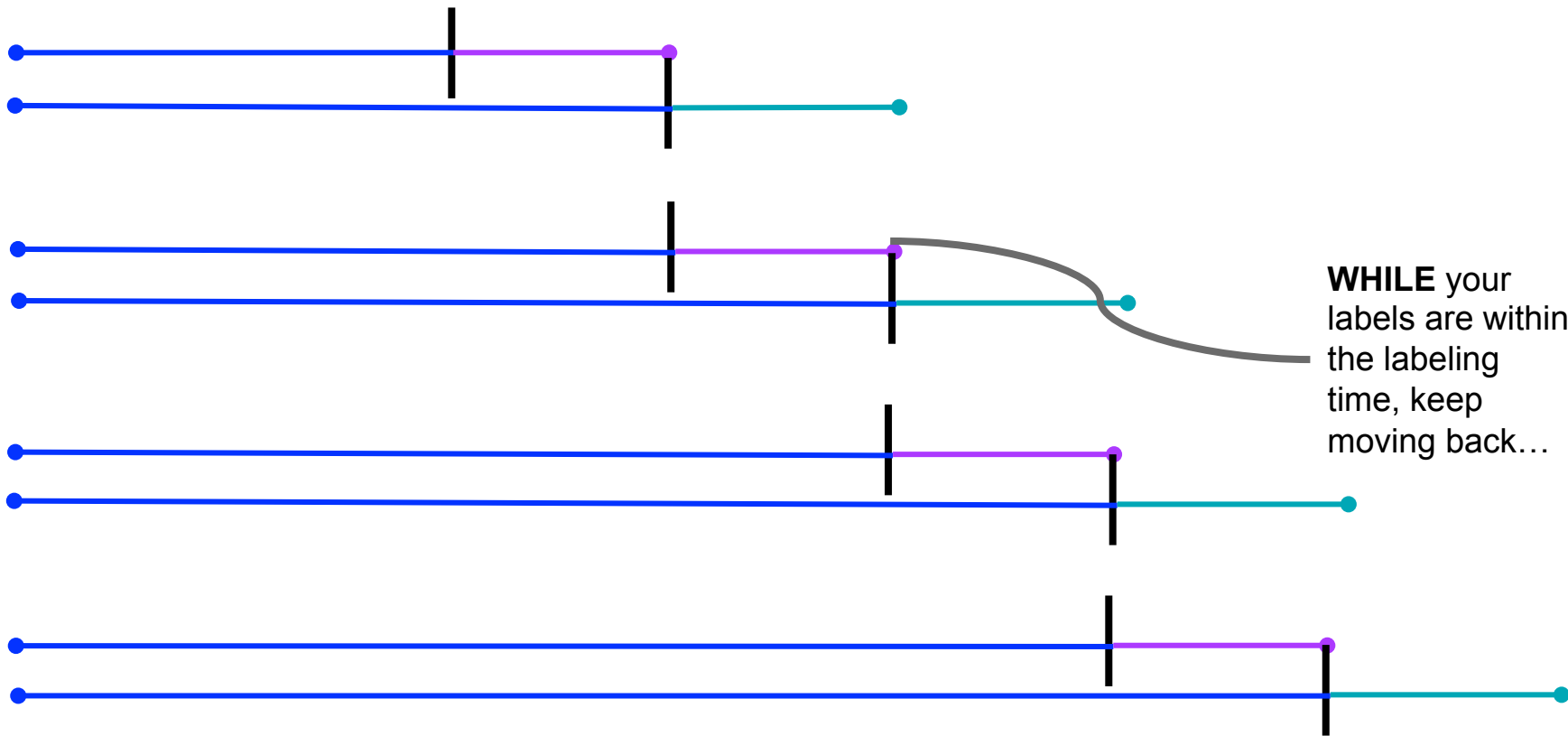
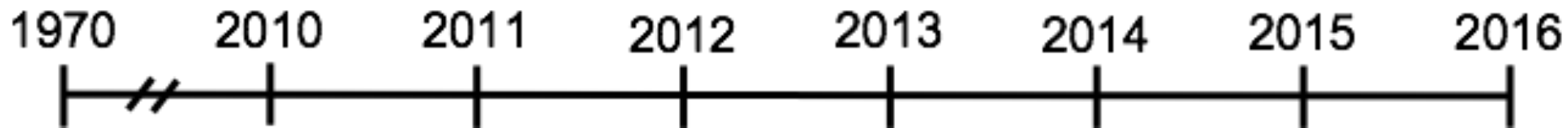


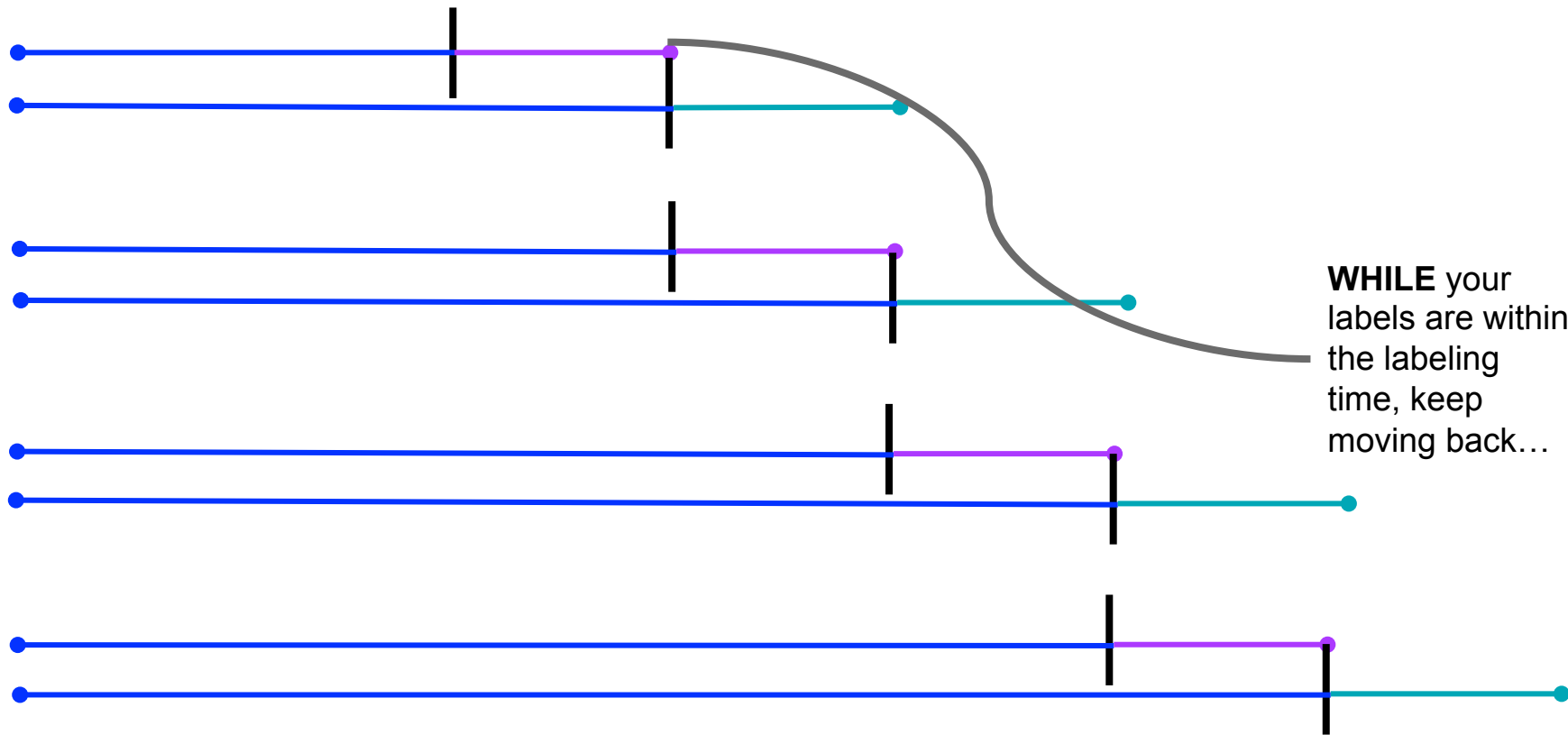
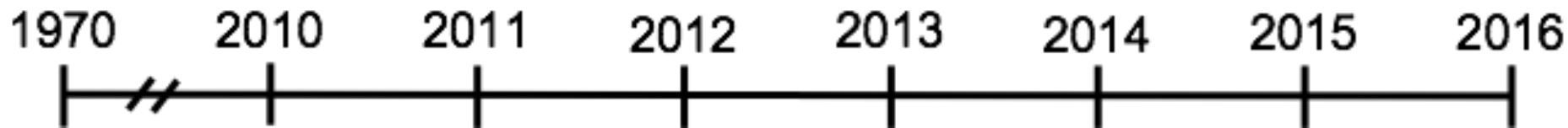


**Use the
freshest data:**
Start from the
last split and
roll back

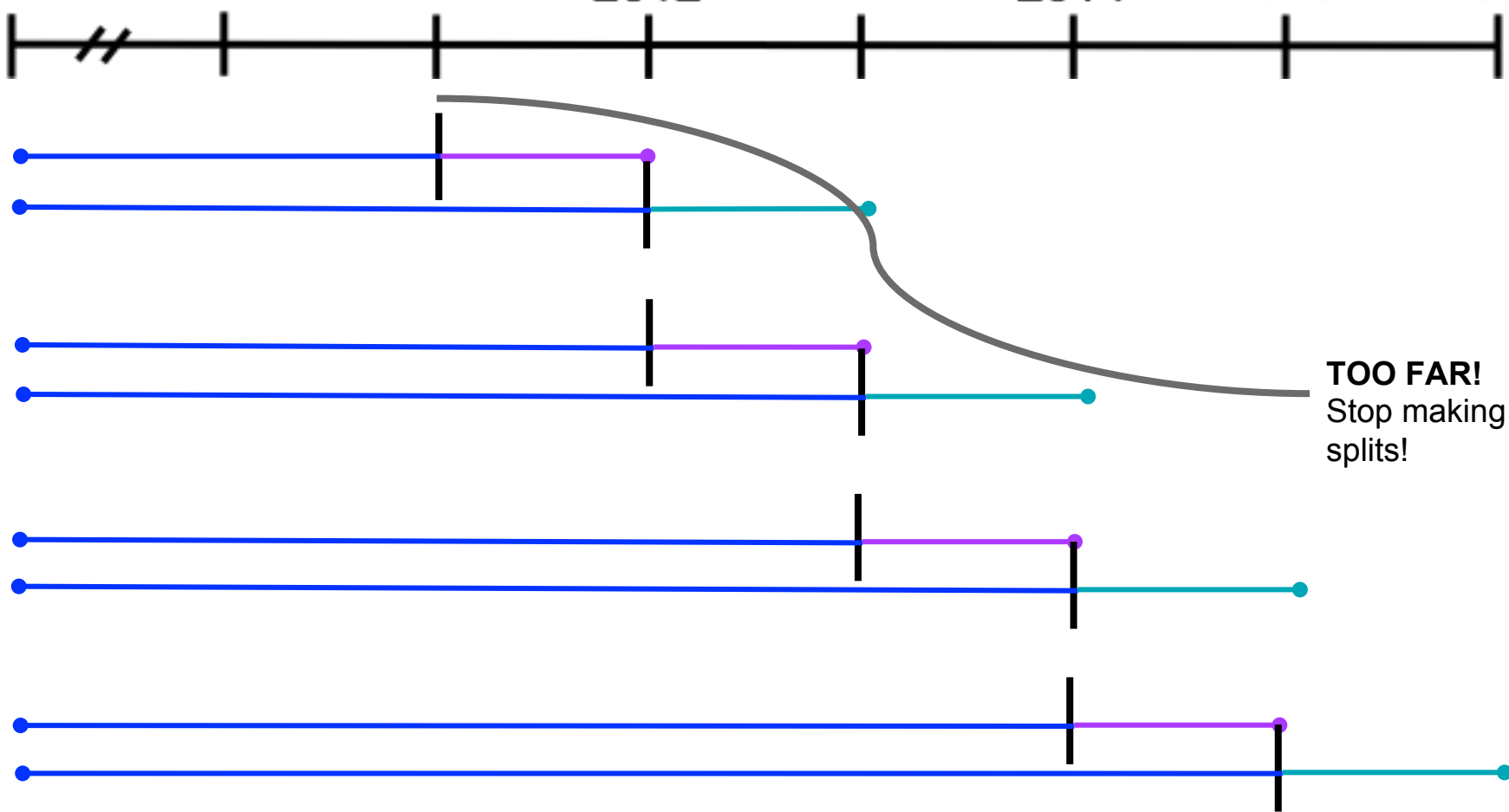


WHILE your labels are within the labeling time, keep moving back...





1970 2010 2011 2012 2013 2014 2015 2016



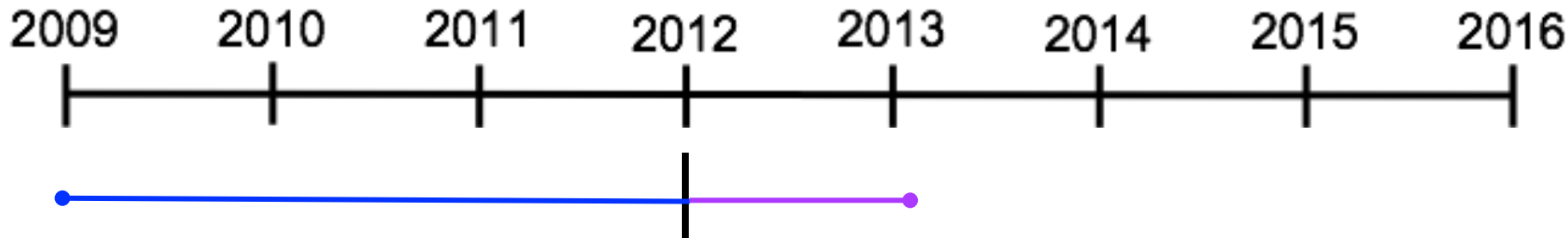
TOO FAR!
Stop making
splits!

Configuring Labels

Labels aggregate data over a fixed time period, beginning at the feature label cut point (as of date), and assign a score.

For example:

- Will this property have a housing code violation in the next year?

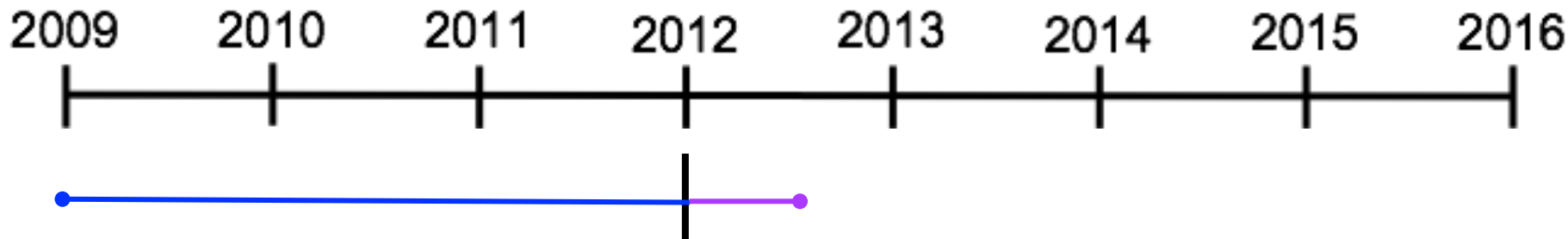


Configuring Labels

Labels aggregate data over a fixed time period, beginning at the feature label cut point, and assign a score.

For example:

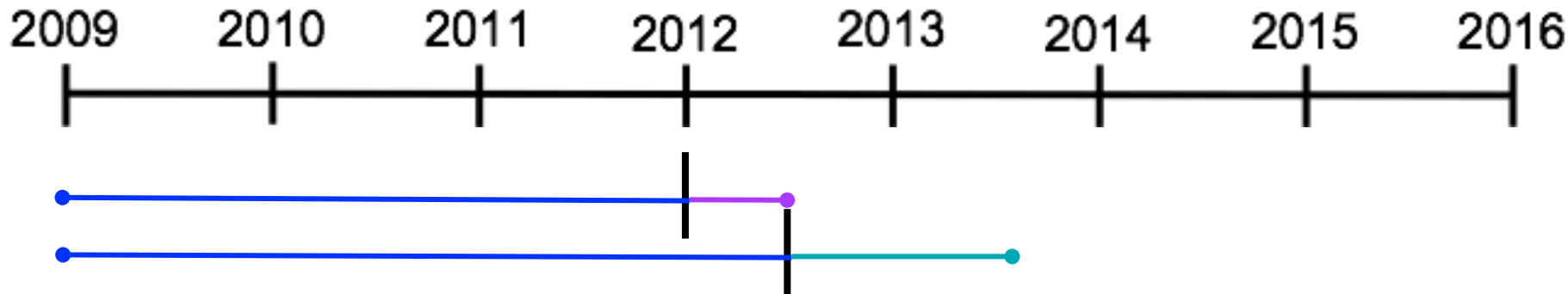
- Will this property have a housing code violation in the next year?
- How many bookings will this person have in the next six months?



Configuring Labels

The length of time aggregated into the label is the **label timespan**. It can be configured separately for train and test labels.

- **train_label_timespan**: how much time is aggregated into labels in the training matrix?
- **test_label_timespan**: how much time is aggregated into labels in the test matrix?



train

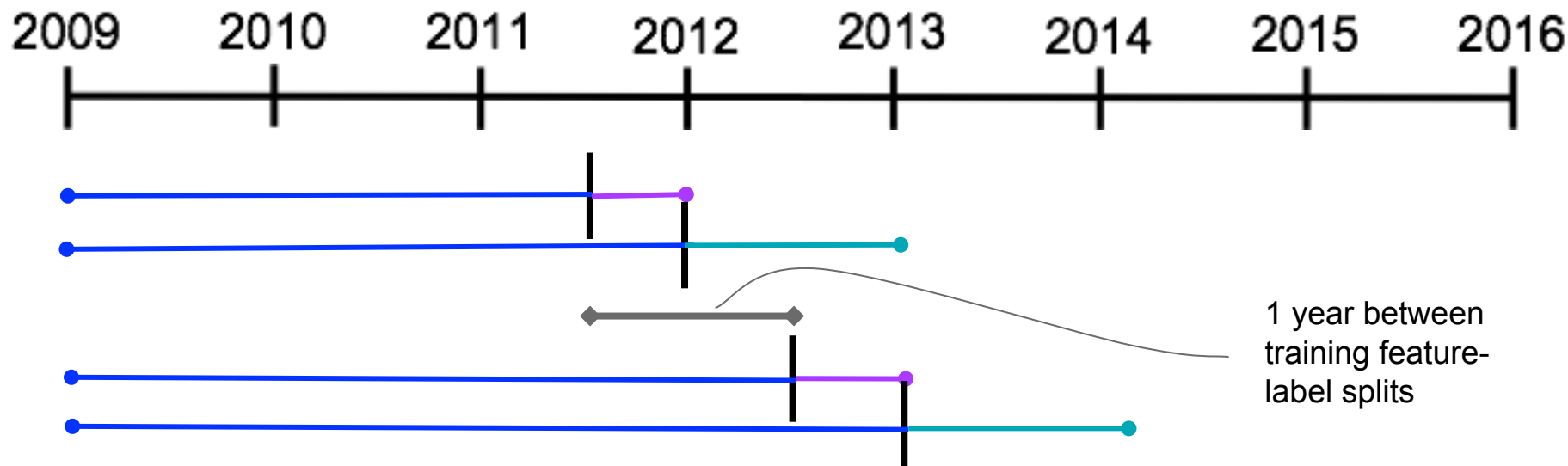
entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-01-01	3	1
2	2012-01-01	1	0
3	2012-01-01	1	1

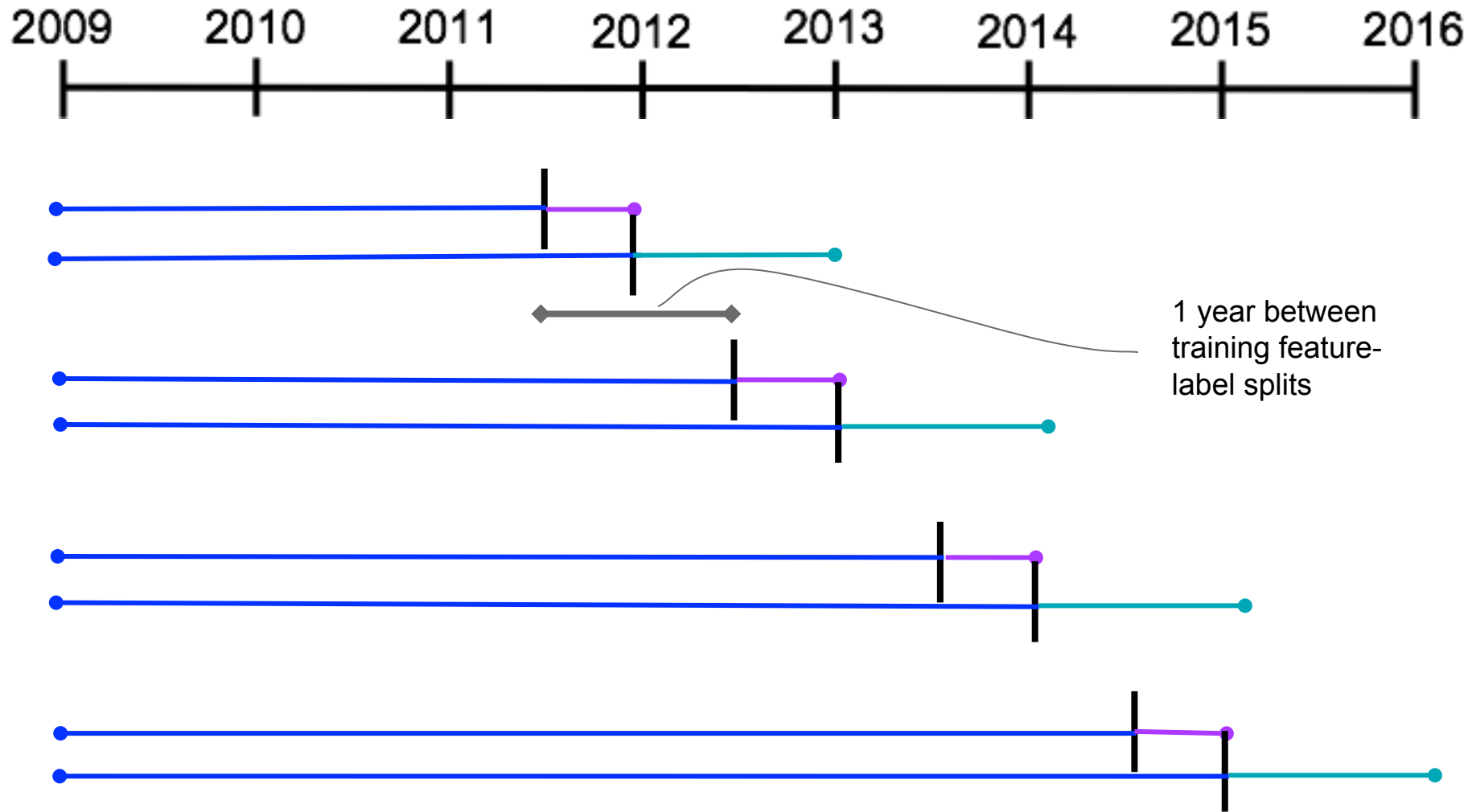
test

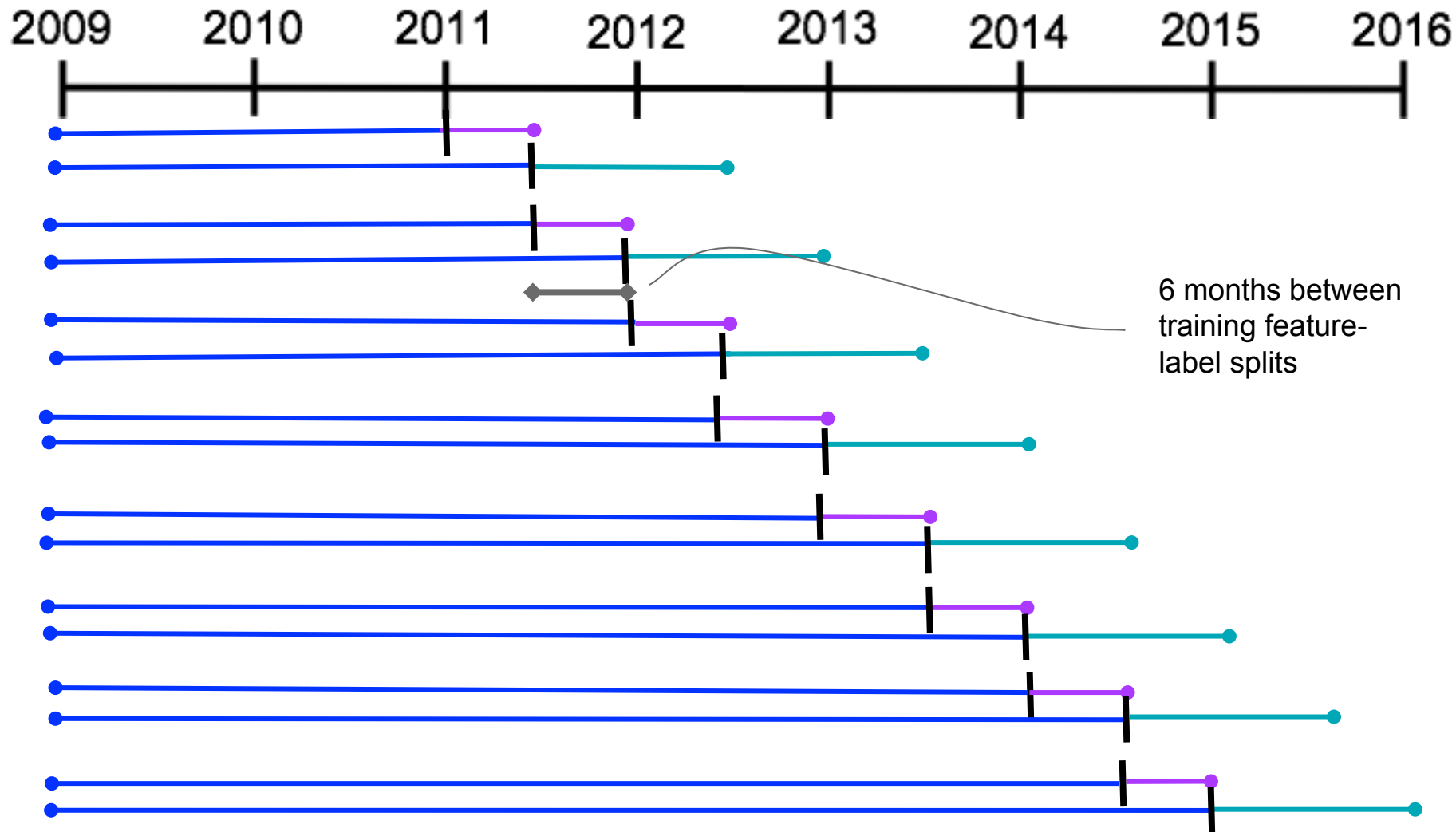
entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-07-01	3	1
2	2012-07-01	1	1
3	2012-07-01	0	1

How often to retrain the model?

You can retrain models after any arbitrary amount of time, called the **model update frequency**. For example...

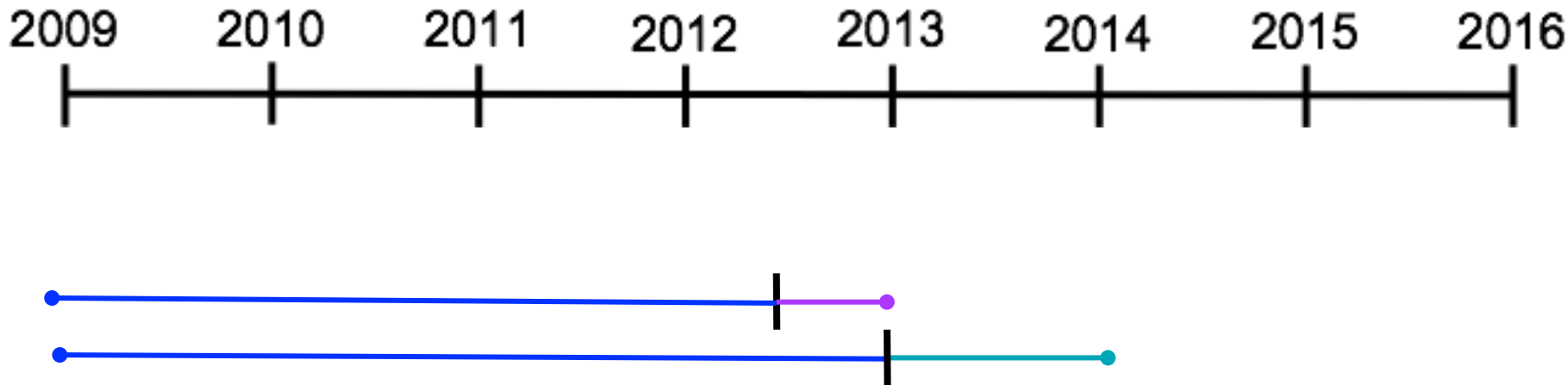






How many rows to put in the matrix?

So far, all of the matrices have one row per entity. We train on one time (July 1, 2012) and test on one time (January 1, 2013).



train

entity_id	as_of_date	bookings_last_4_yrs	label
1	2012-07-01	3	1
2	2012-07-01	1	0
3	2012-07-01	1	1

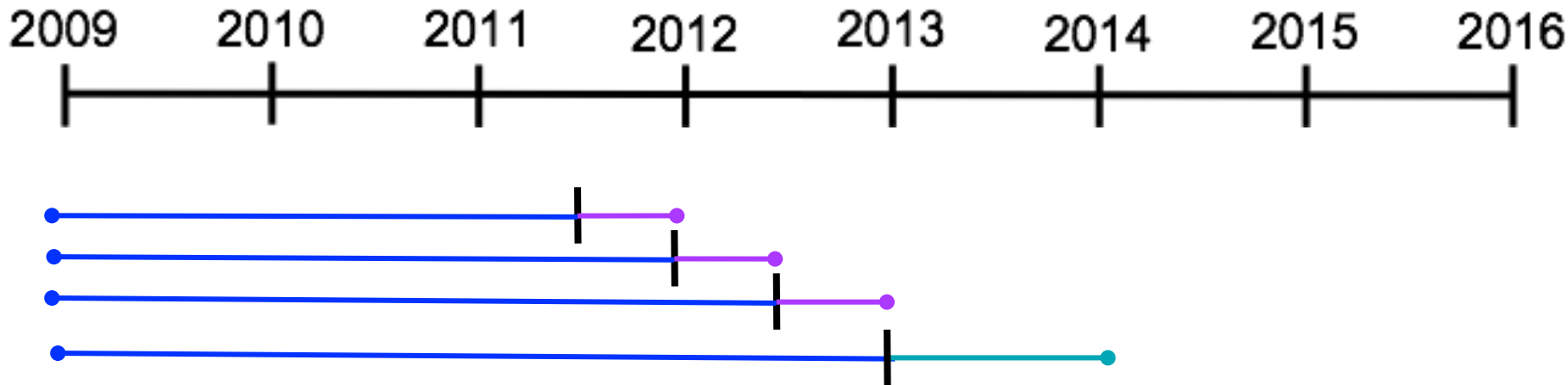
test

entity_id	as_of_date	bookings_last_4_yrs	label
1	2013-01-01	3	1
2	2013-01-01	1	1
3	2013-01-01	0	1

Temporal “oversampling”

So far, all of the matrices have one row per entity. We train on one time (January 1, 2012) and test on one time (July 1, 2012).

But we can train the **same model** on multiple dates....



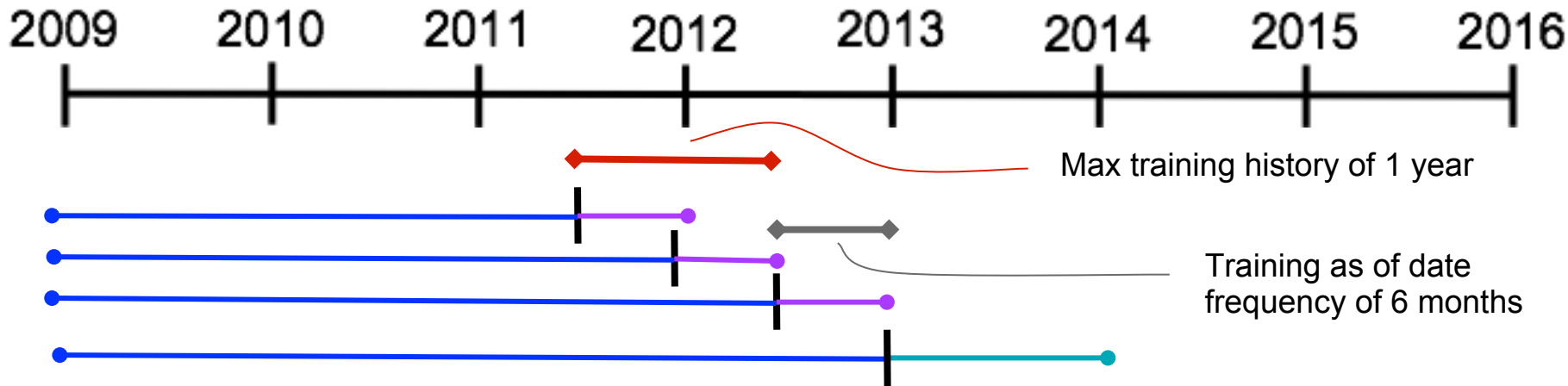
train

entity_id	as_of_date	bookings_last_4_yrs	label
1	2011-07-01	3	1
1	2012-01-01	2	1
1	2012-07-01	4	1
2	2011-07-01	0	1
2	2012-01-01	1	0
2	2012-07-01	0	0
3	2011-07-01	1	1
3	2012-01-01	1	0
3	2012-07-01	2	1

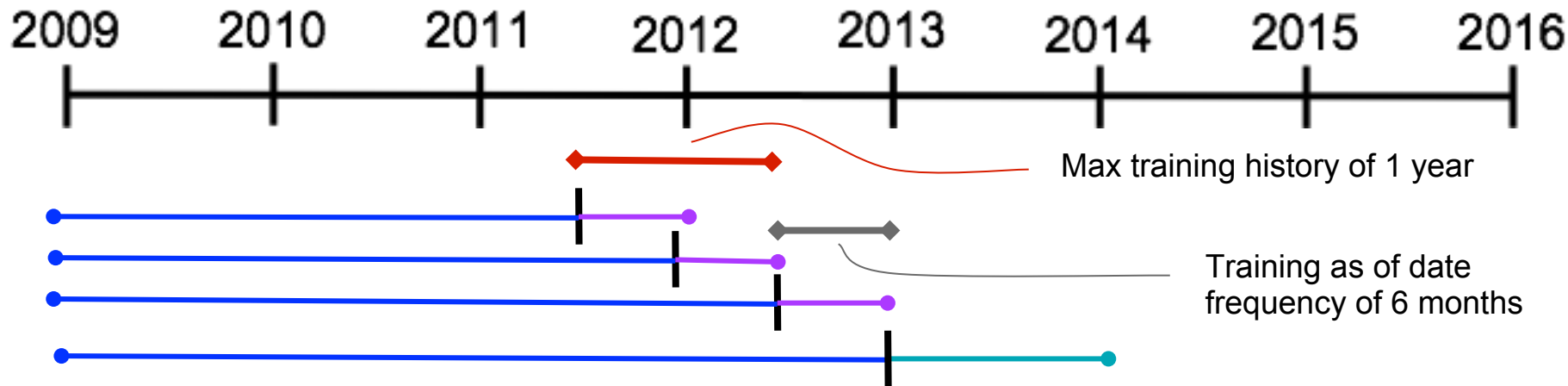
Configuring Temporal Parameters

How many as of dates are in a training matrix depends on 2 parameters:

- **training_as_of_date_frequency**: how much time between dates
- **max_training_history**: how much time between first and last training as of dates (at most)



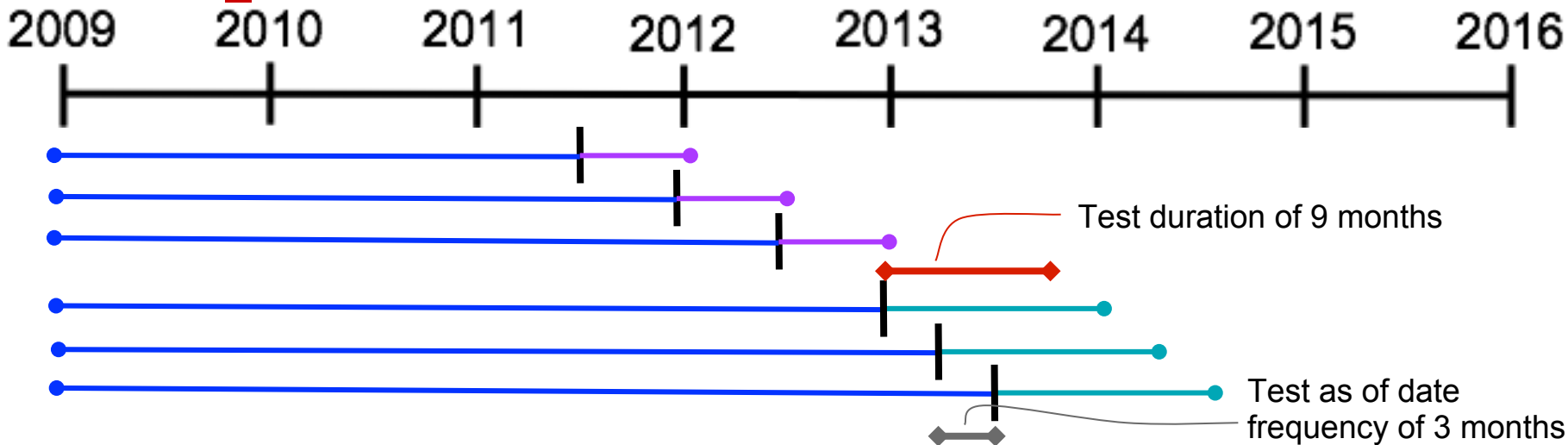
```
last_as_of_date = train_test_split_date - train_label_timespan
earliest_possible_as_of_date = last_as_of_date - max_training_history
as_of_dates = []
current_as_of_date = last_as_of_date
WHILE current_as_of_date >= earliest_possible_as_of_date:
    as_of_dates.append(current_as_of_date)
    current_as_of_date = current_as_of_date - as_of_date_frequency
```



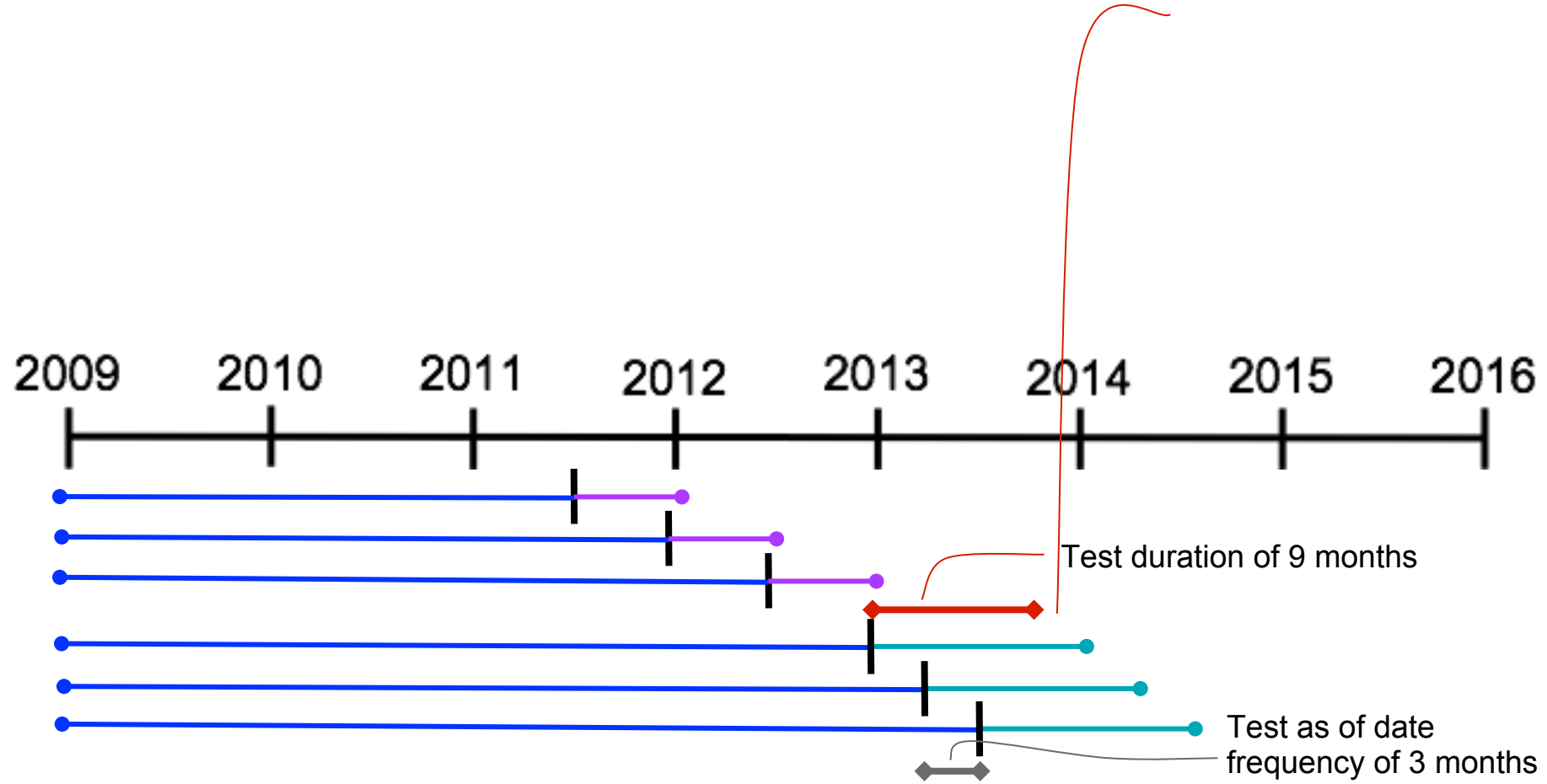
Configuring Temporal Parameters

The same controls exist for test matrices, allowing you to include multiple test dates for the same model:

- **test_as_of_date_frequency**: how much time between dates
- **test_duration**: how much time between first and last test as of dates



Same loop, but roll **forward** this time, and don't forget to **always use <**



test

entity_id	as_of_date	bookings_last_4_yrs	label
1	2013-01-01	3	1
1	2013-04-01	3	1
1	2013-07-01	4	1
2	2013-01-01	0	1
2	2013-04-01	1	0
2	2013-07-01	0	0
3	2013-01-01	1	1
3	2013-04-01	3	0
3	2013-07-01	3	1

All the Temporal Parameters

temporal_config:

feature_start_time: '1995-01-01' # earliest date included in features

feature_end_time: '2015-01-01' # latest date included in features

label_start_time: '2012-01-01' # earliest date for which labels are available

label_end_time: '2015-01-01' # day AFTER last label date (all dates in any model are < this date)

model_update_frequency: ['6month'] # how frequently to retrain models

training_as_of_date_frequencies: ['1day'] # time between as of dates for same entity in train matrix

test_as_of_date_frequencies: ['3month'] # time between as of dates for same entity in test matrix

max_training_histories: ['6month', '3month'] # length of time included in a train matrix

test_durations: ['0day', '1month', '2month'] # length of time included in a test matrix

training_label_timespans: ['1month'] # time period across which outcomes are labeled in train matrices

test_label_timespans: ['7day'] # time period across which outcomes are labeled in test matrices

General Structure

- **Outer loop:** Find train-test split dates within label and feature time
 - Roll backwards from last possible split to use your freshest data
- **Inner loops:** Find feature-label split dates (as-of-dates) with the matrix duration
 - Train matrices: roll backward from the latest possible as-of-date
 - Test matrices: roll forward from the train-test split date

Warnings

- ALWAYS USE \geq AND \leq
- Test your temporal code: Start with a simple configuration where **you can write out the expected results** without code, and make sure your code produces the right dates
- Test your feature code: Start with a few people and a few features that you made up, calculate the expected matrices by hand and check them against the matrices your code makes

Sources of Temporal Leakage

Obvious:

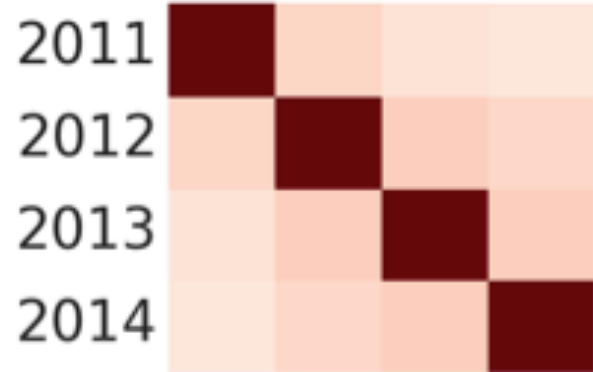
- Train and test labels aggregate data from overlapping times
- Labels and features aggregate data from overlapping times

Less obvious:

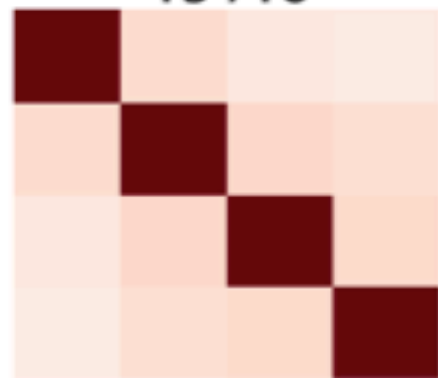
- Cohorts: People not known in the data until later are included in earlier models

Temporal Model Evaluation

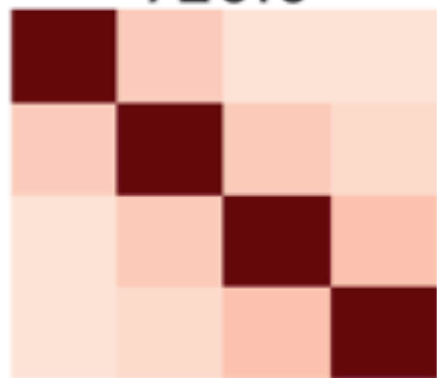
295.0



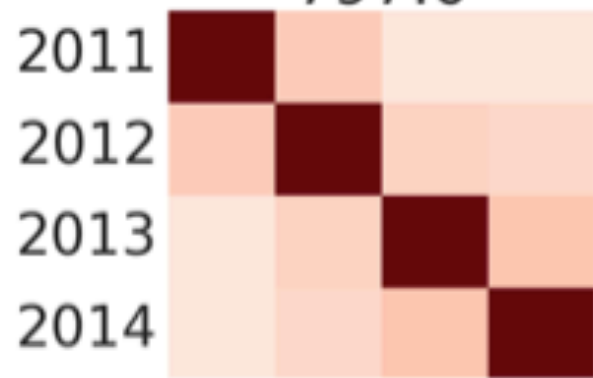
497.0



720.0

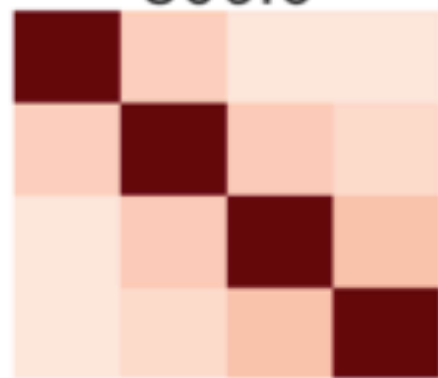


797.0



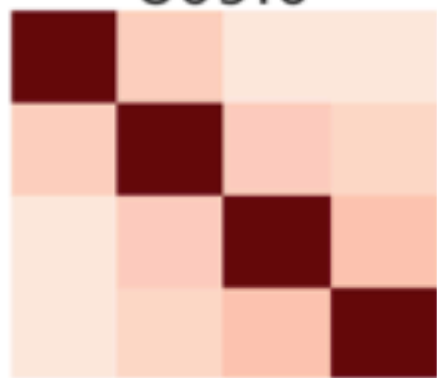
2011 2012 2013 2014

806.0



2011 2012 2013 2014

809.0



2011 2012 2013 2014

1.0

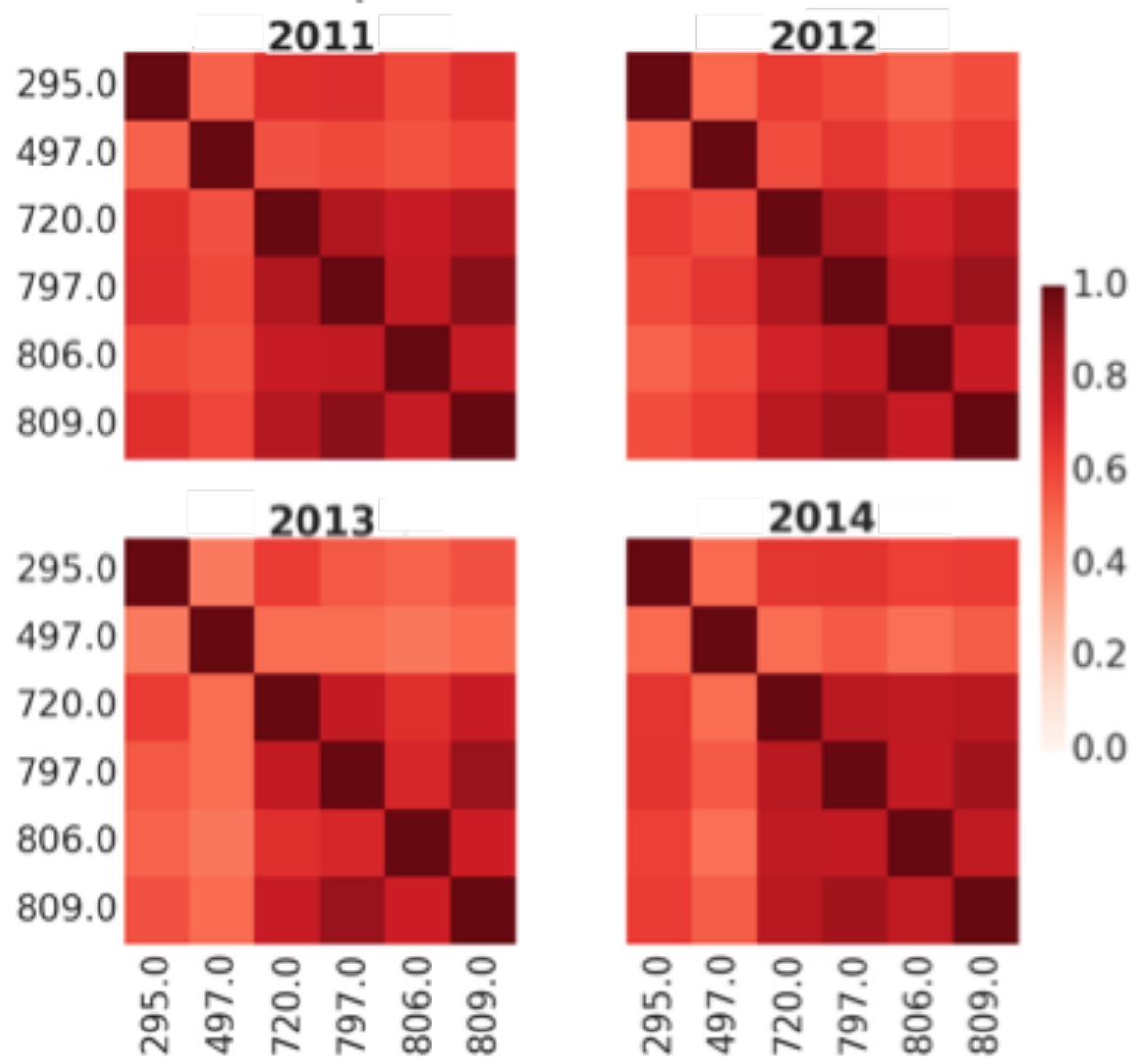
0.8

0.6

0.4

0.2

0.0



FEATURE IMPORTANCE RANK CORRELATION, WITHIN MODELS

