Linear trend: using time as a feature

Trend features

Linear trend

Let's consider a linear model:

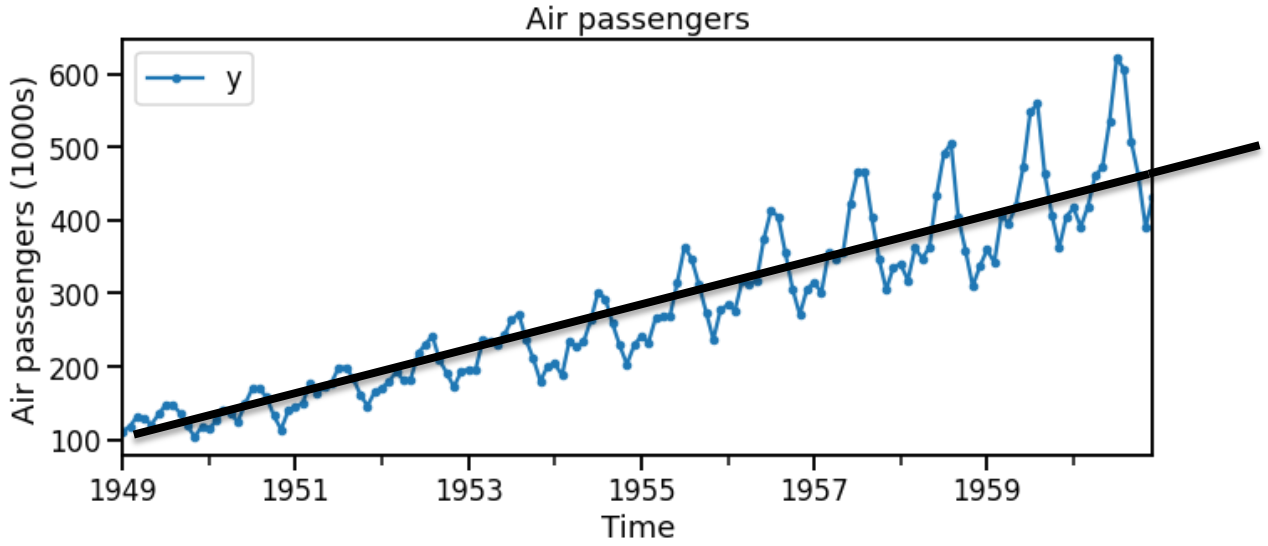
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$$

Model linear trend using time passed since a reference time, t_0, as feature $(t - t_0)$:

$$y_t = \beta_0 + \beta_1 (t - t_0)$$

We typically set t_0 to the start time of the time series $t_0 = 0$:

$$y_t = \beta_0 + \beta_1 t$$ ← This feature can capture trend.



Creating the feature for training and prediction

Time	Daily Sales
2020-02-12	23
2020-02-13	30
2020-02-14	35
2020-02-15	30
2020-02-16	?
2020-02-17	?
2020-02-18	?

Creating the feature for training and prediction

Time	Daily Sales	t (days)
2020-02-12	23	0
2020-02-13	30	1
2020-02-14	35	2
2020-02-15	30	3
2020-02-16	?	4
2020-02-17	?	5
2020-02-18	?	6

Creating the feature for training and prediction

Time	Daily Sales	t (days)
2020-02-12	23	0
2020-02-13	30	1
2020-02-14	35	2
2020-02-15	30	3
2020-02-16	?	4
2020-02-17	?	5
2020-02-18	?	6

Note gaps in the time stamp

Time	Monthly Sales
2020-02-01	23
2020-03-01	30
2020-05-01	35
2020-07-01	30
2020-08-01	?
2020-09-01	?
2020-10-01	?

Creating the feature for training and prediction

Time	Daily Sales	t (days)
2020-02-12	23	0
2020-02-13	30	1
2020-02-14	35	2
2020-02-15	30	3
2020-02-16	?	4
2020-02-17	?	5
2020-02-18	?	6

Note gaps in the time stamp

Time	Monthly Sales	t (months)
2020-02-01	23	0
2020-03-01	30	1
2020-05-01	35	3
2020-07-01	30	5
2020-08-01	?	6
2020-09-01	?	7
2020-10-01	?	8

Forecasting with just the time feature

	Target **y**	Features **X**
Time	**y**	***t*** **(days)**
2020-02-12	23	0
2020-02-13	30	1
2020-02-14	35	2
2020-02-15	30	3
2020-02-16		4
2020-02-17		5
2020-02-18		6

model.fit(X_train, y_train)

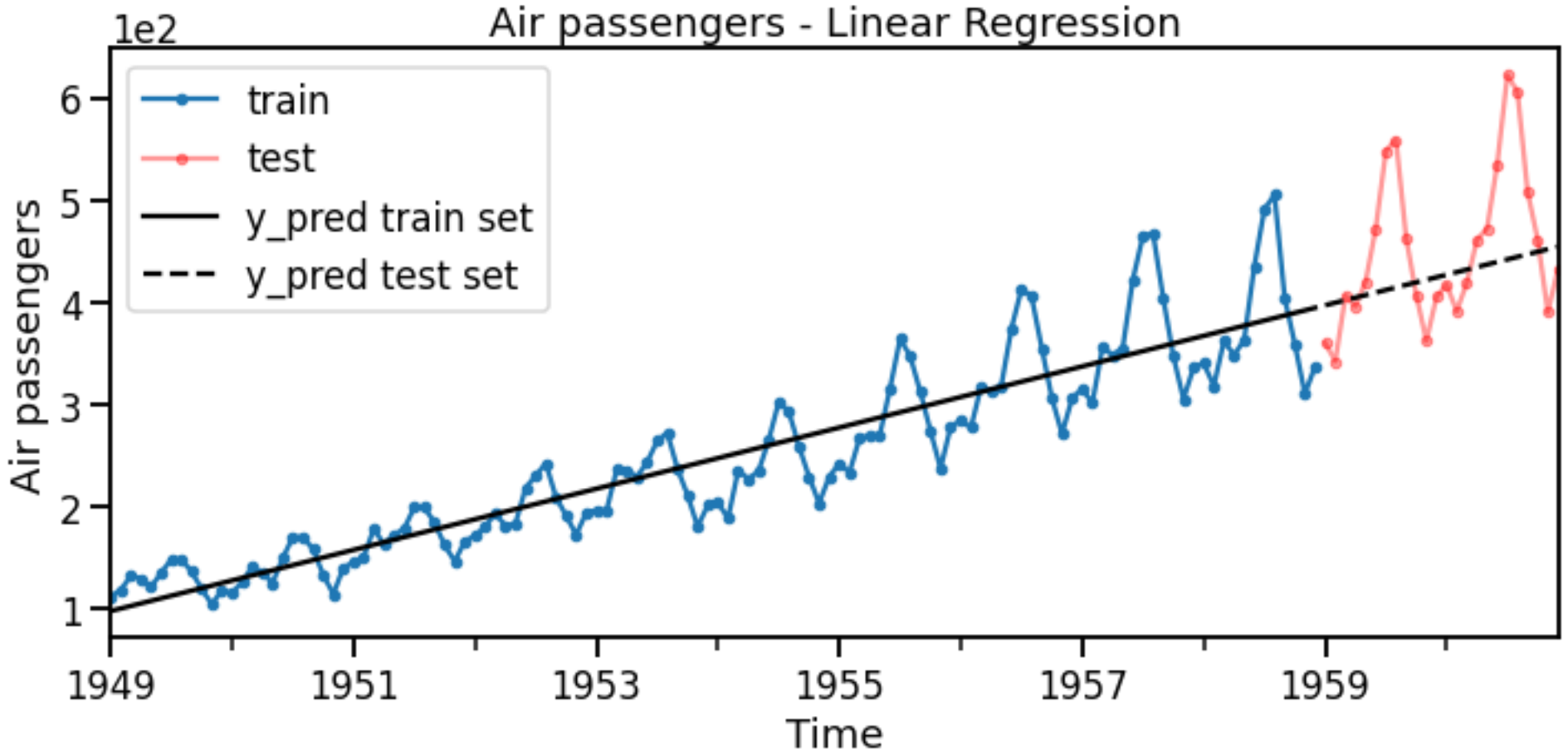
model.predict $\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \rightarrow \begin{pmatrix} \hat{y}_{T+1} \\ \hat{y}_{T+2} \\ \hat{y}_{T+3} \end{pmatrix}$

Forecasting with just the time feature

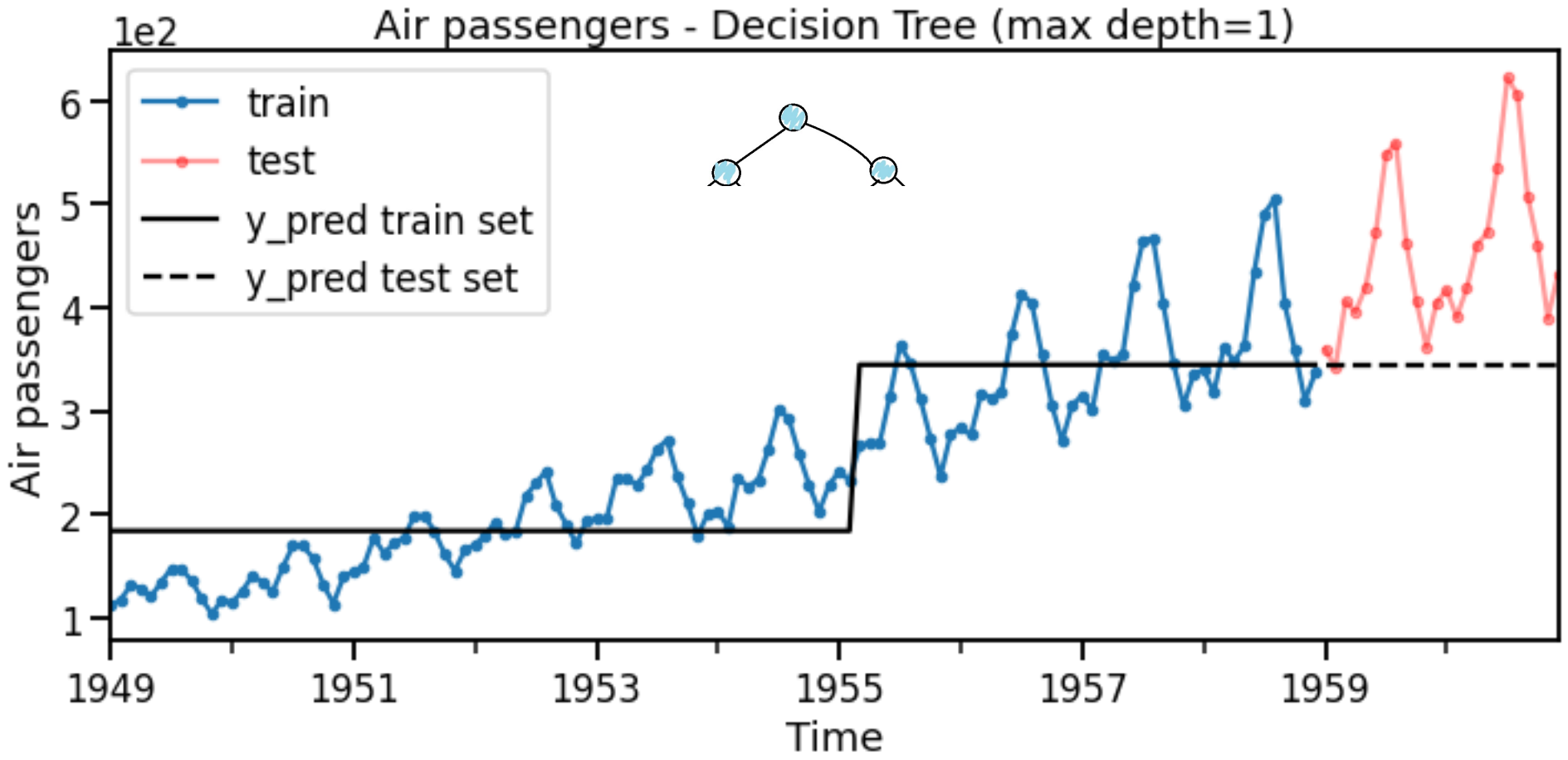
	Target **y**	Features **X**
Time	**y**	t **(days)**
2020-02-12	23	0
2020-02-13	30	1
2020-02-14	35	2
2020-02-15	30	3
2020-02-16	$\hat{y}_{T+1}$	4
2020-02-17	$\hat{y}_{T+2}$	5
2020-02-18	$\hat{y}_{T+3}$	6

model.predict $\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$ → $\begin{bmatrix} \hat{y}_{T-3} \\ \hat{y}_{T-2} \\ \hat{y}_{T-1} \\ \hat{y}_T \end{bmatrix}$

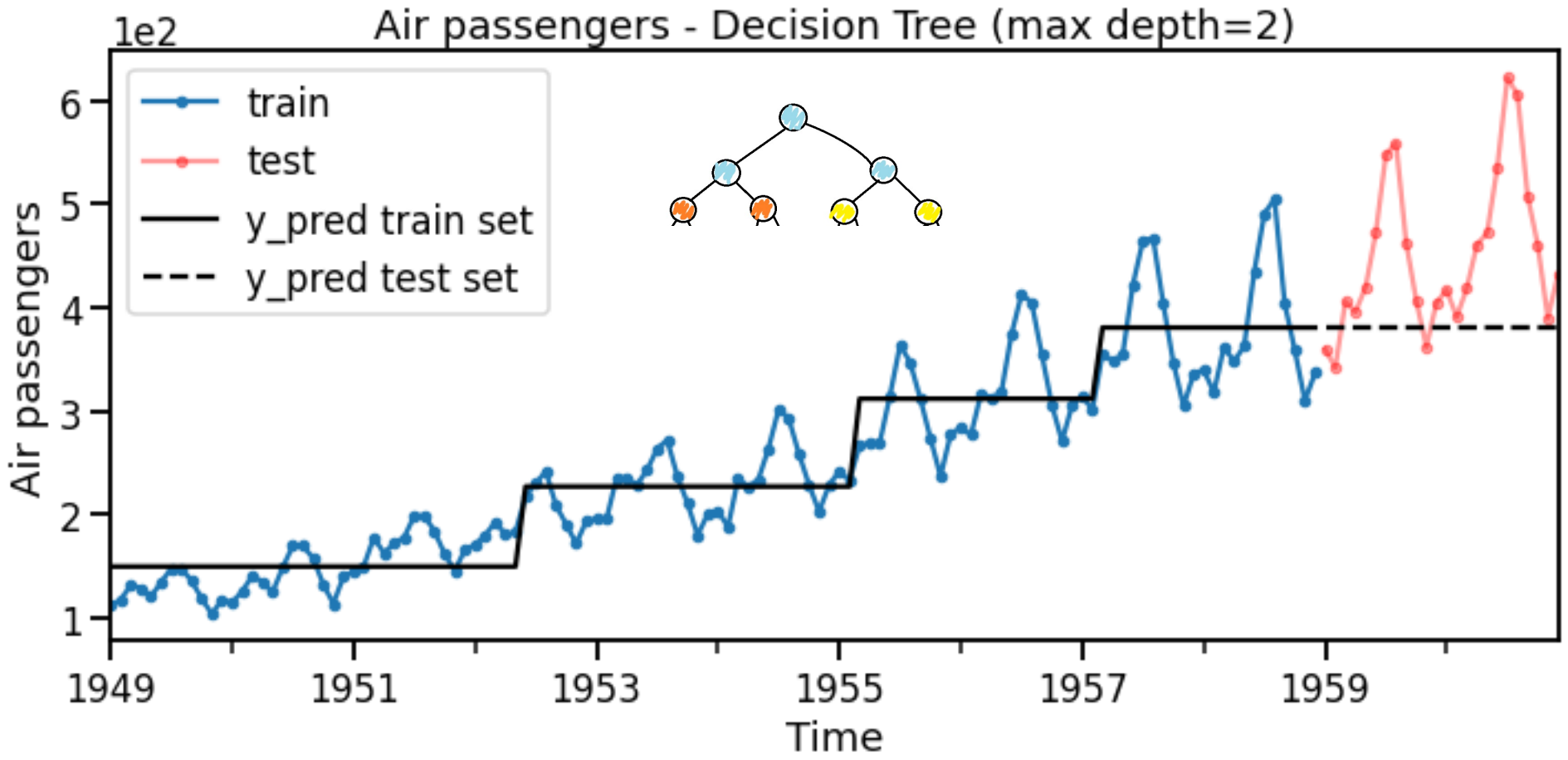
Example: Linear regression with time feature



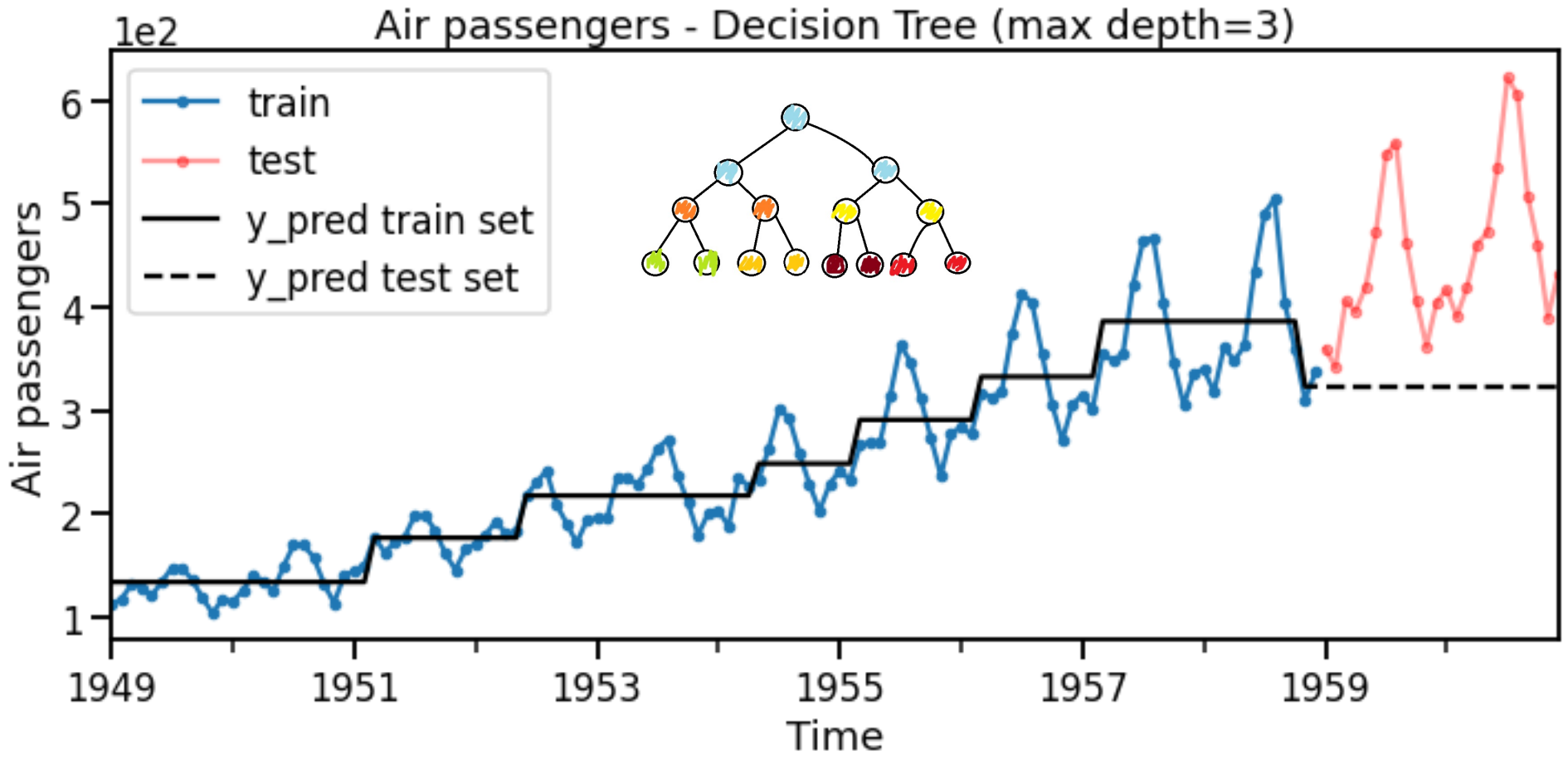
Example: Tree-based models with time feature



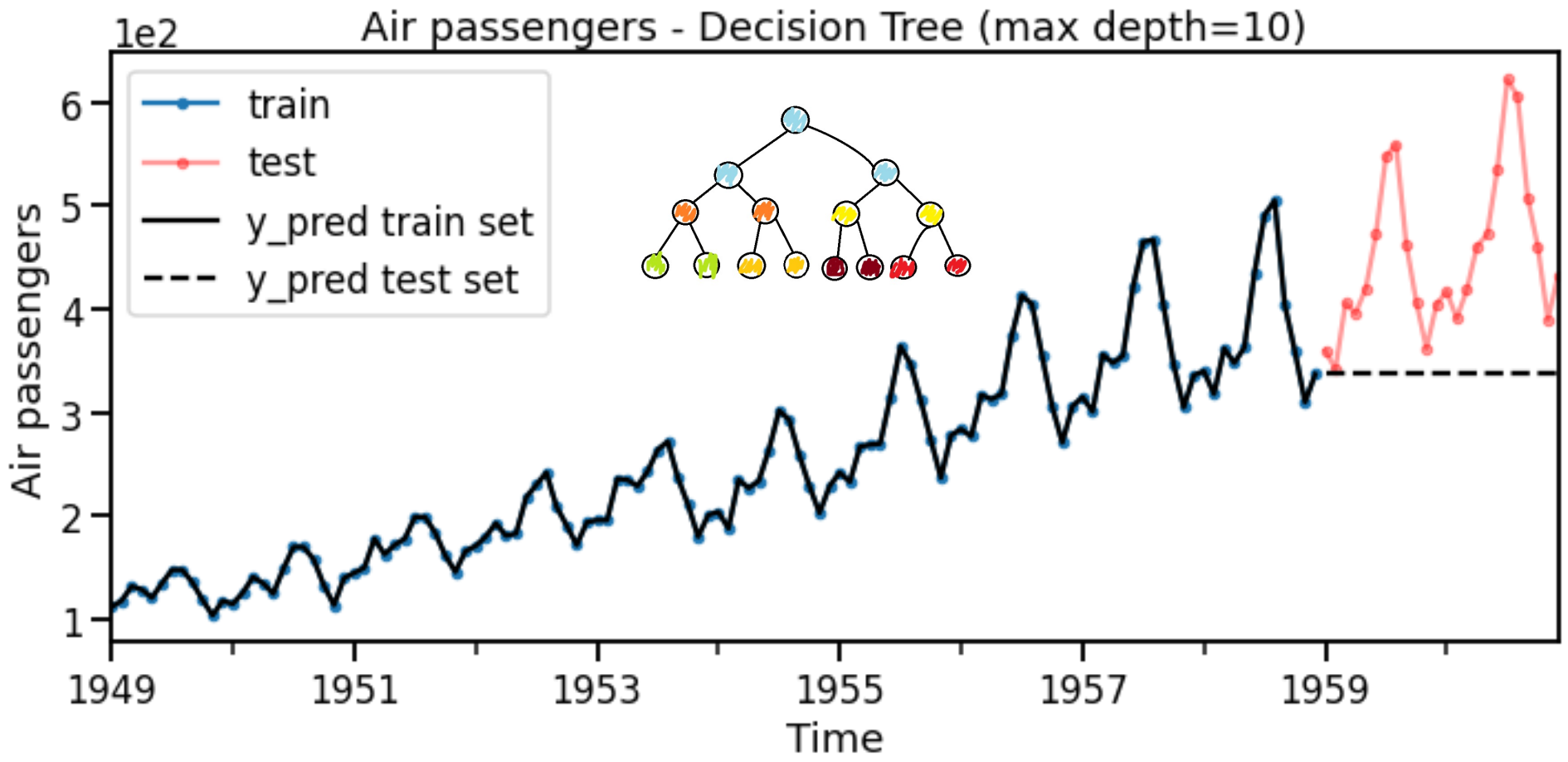
Example: Tree-based models with time feature



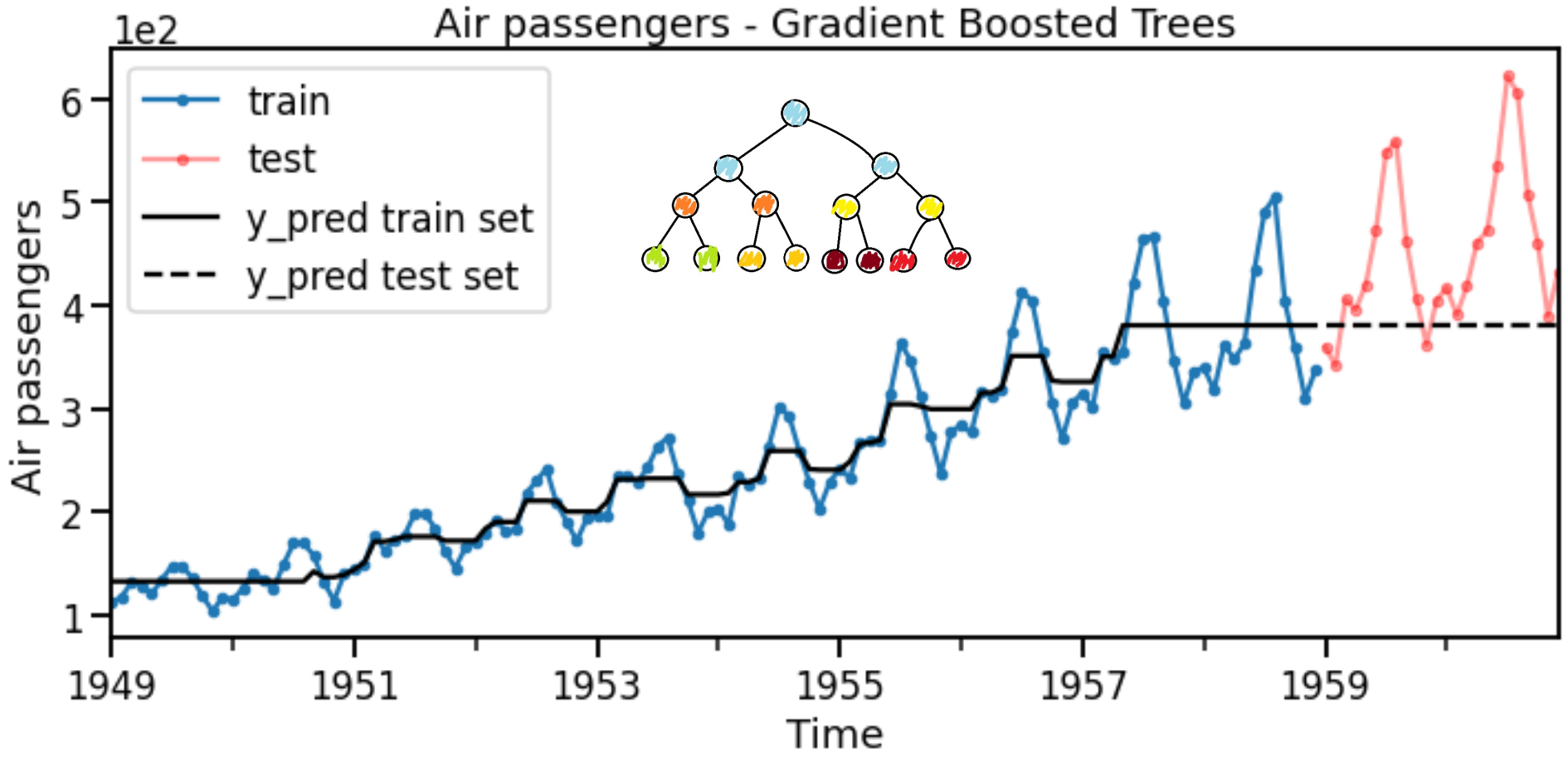
Example: Tree-based models with time feature



Example: Tree-based models with time feature



Example: Tree-based models with time feature



Implementation – Pandas & Numpy

```
df["t"] = np.round((df.index - df.index.min()) / np.timedelta64(1 , "M"))
df.head()
```

ds	y	t
1949-01-01	112	0.0
1949-02-01	118	1.0
1949-03-01	132	2.0
1949-04-01	129	3.0
1949-05-01	121	4.0

Implementation - sktime

```
from sktime.transformations.series.time_since import TimeSince
```

```
transformer = TimeSince(start=["1949-01-01", "1949-02-01"], # A list of start dates.
                                                            # If `None`, uses earliest time in dataframe.
                        to_numeric=True, # Convert output to integer or keep as time-like.
                        freq="MS", # Specify time series frequency if not specified in dataframe.
                        positive_only=False, # Set negative values to zero.
                        keep_original_columns=False, # Keep the other columns in the dataframe
                                                     # after passing to `.transform()`.
                       )
```

Implementation - sktime

```
transformer.transform(df)
```

ds	time_since_1949-01-01 00:00:00	time_since_1949-02-01 00:00:00
1949-01-01	0	-1
1949-02-01	1	0
1949-03-01	2	1
1949-04-01	3	2
1949-05-01	4	3

Summary

We can use the time elapsed, t, to model the trend.

In a linear model, using t, results in a linear trend.

Standard tree-based models will not be able to use this feature to extrapolate.