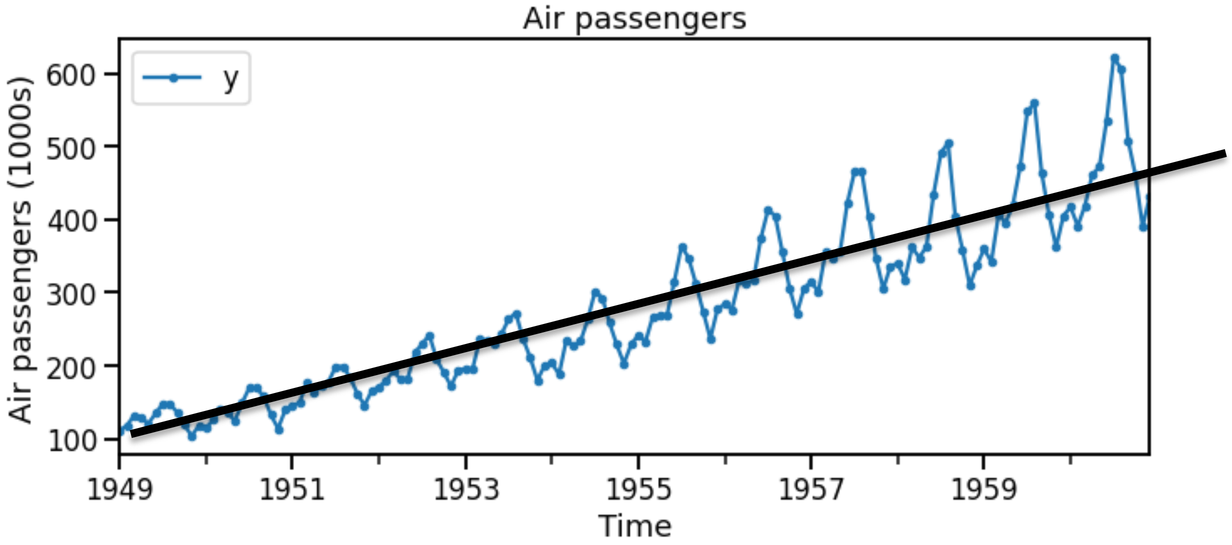
Non-linear trends: using time as a feature

Trend features

Non-linear trends with linear models

Consider the model:

$$y_t = \beta_0 + \beta_1 t$$

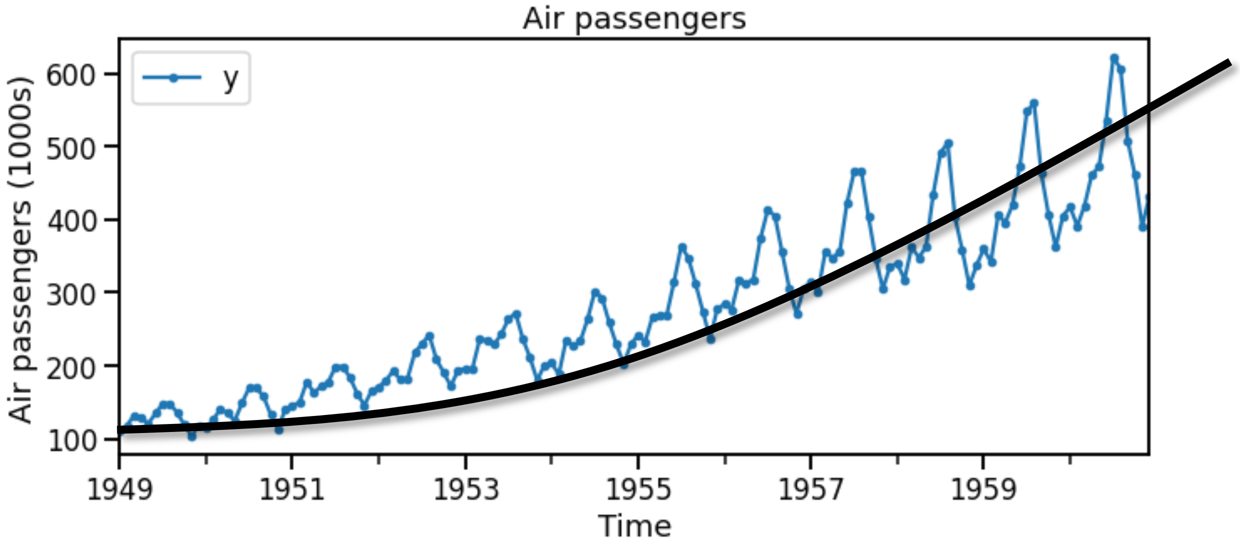


Non-linear trends with linear models

Consider the model:

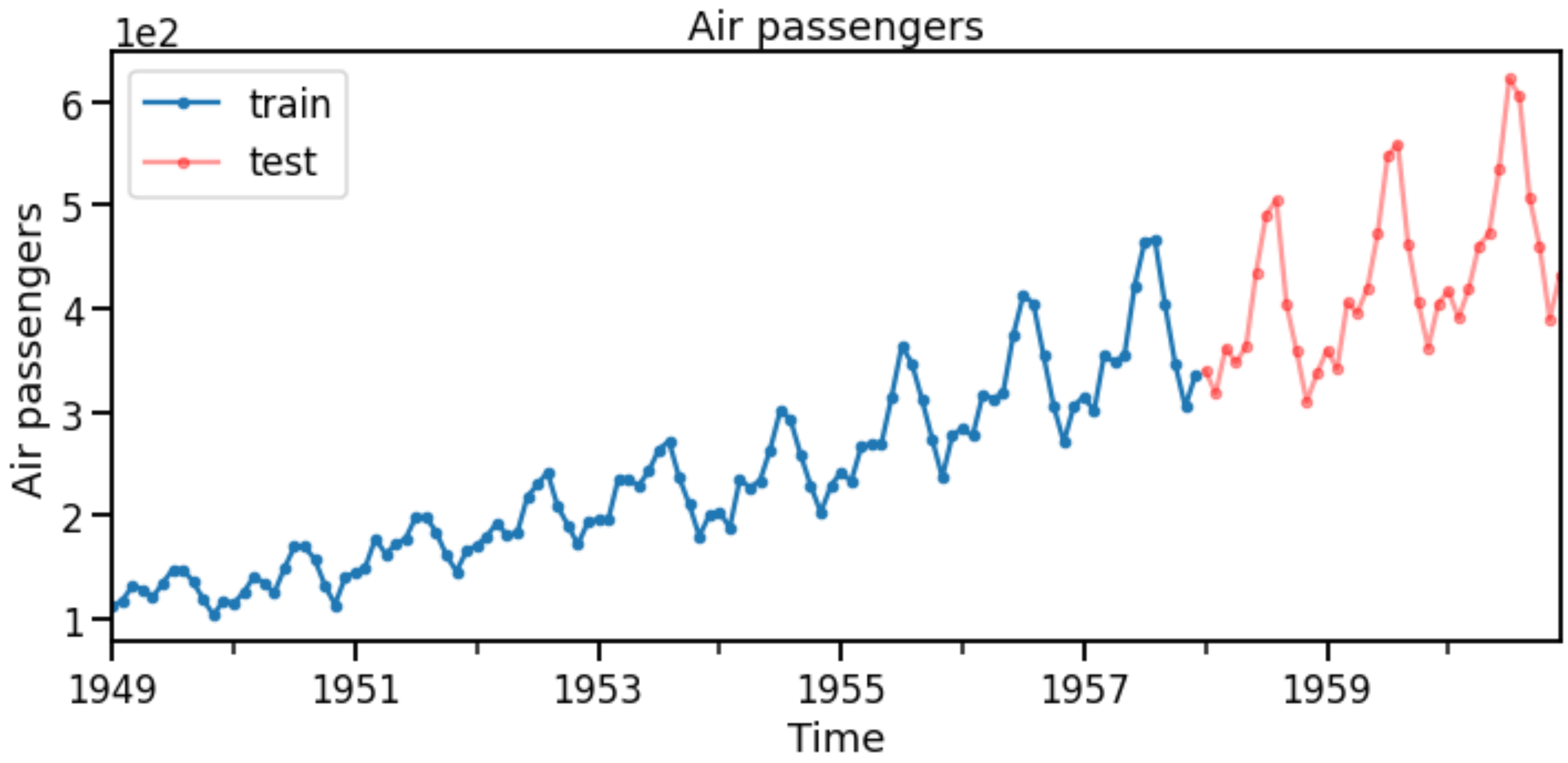
$$y_t = \beta_0 + \beta_1 t + \beta_2 t^2$$

- "When … extrapolated, the resulting forecasts are often unrealistic." [1]
- Risk of overfitting to the training data and extrapolating poorly.
- Piecewise linear trend is recommended instead for non-linear trends. [1]
- Alternative: try regularizing (e.g., Ridge) to limit overfitting.

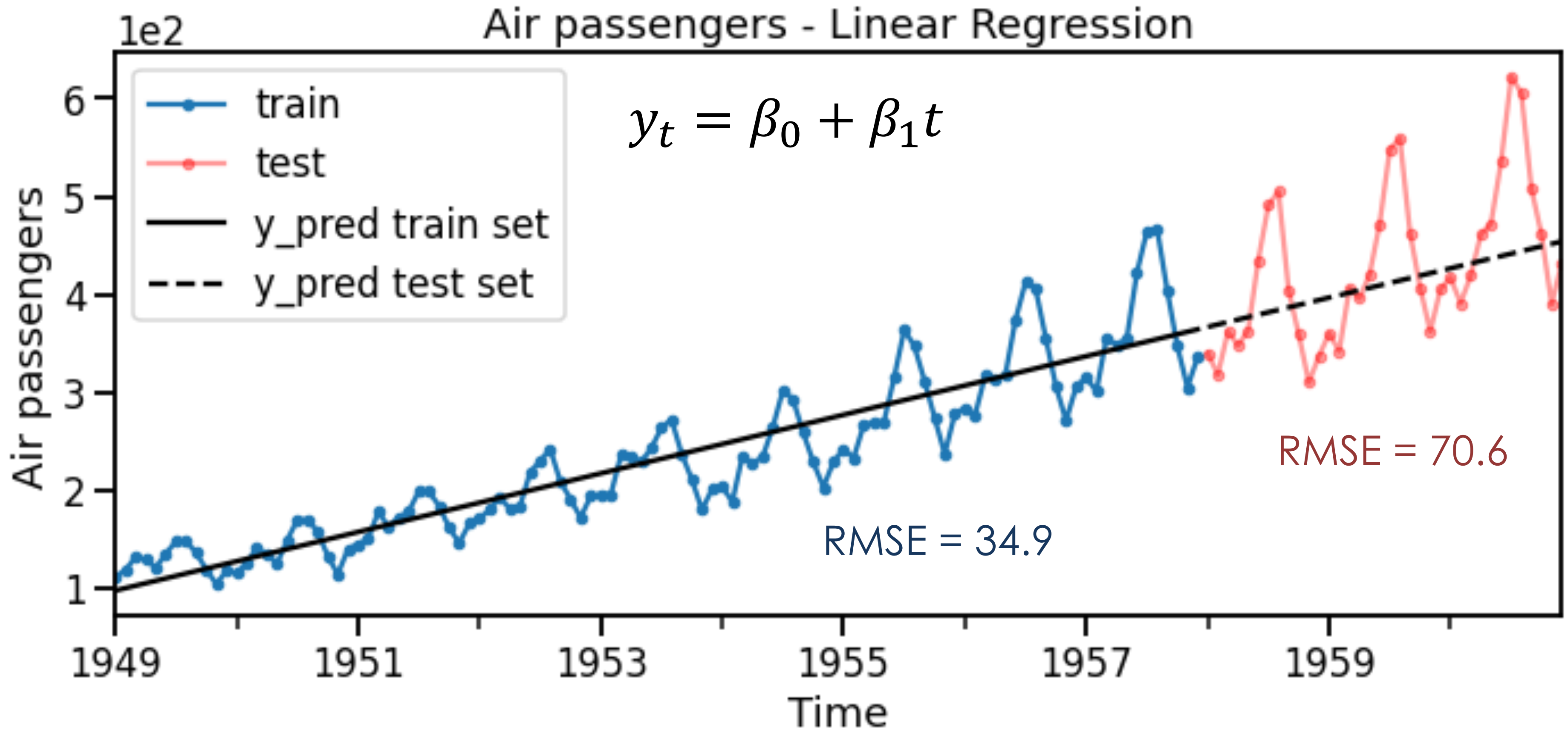


[1] - Hyndman, R.J., & Athanasopoulos, G. (2021) Forecasting: principles and practice, 3rd edition, OTexts: Melbourne, Australia. OTexts.com/fpp3

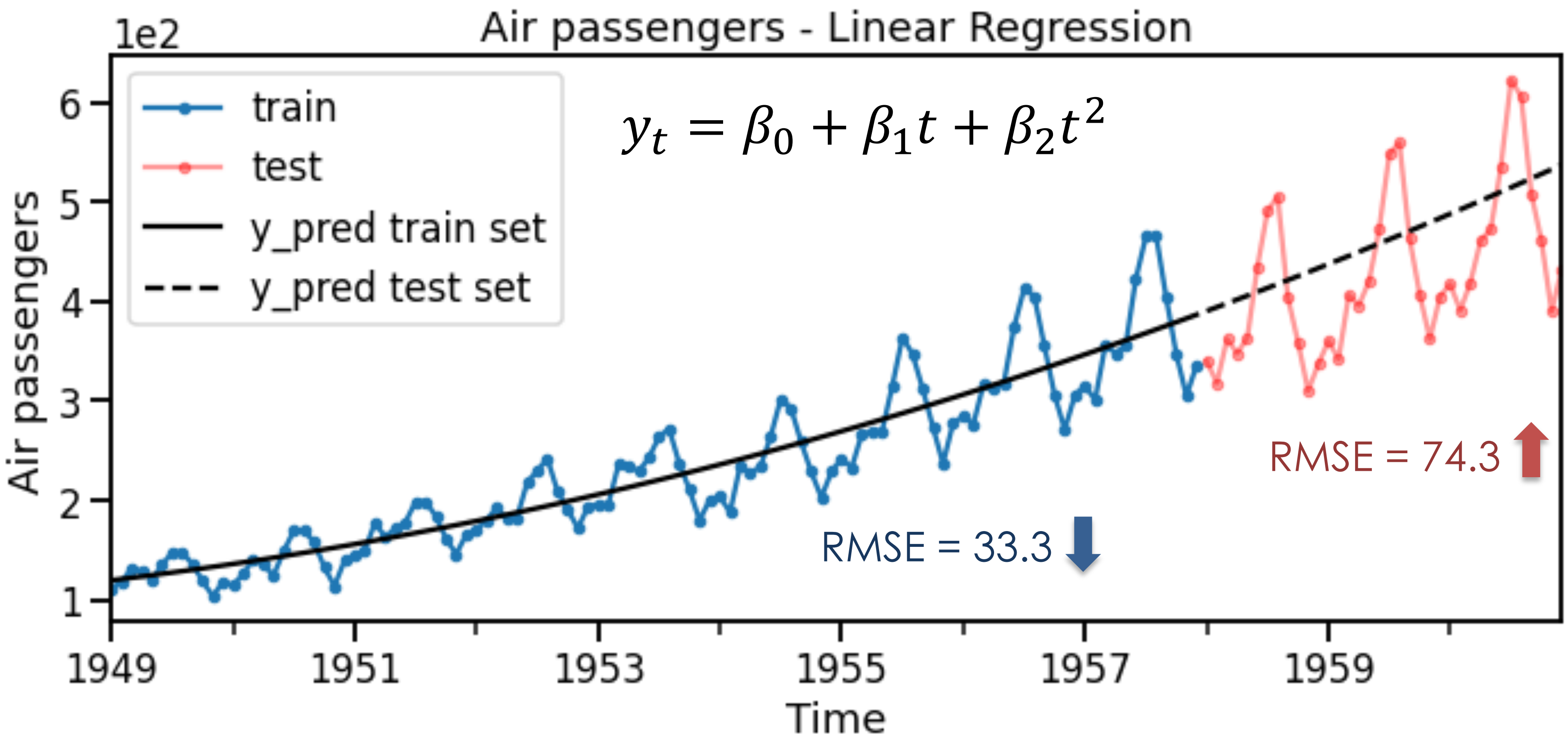
Example: Air passengers dataset



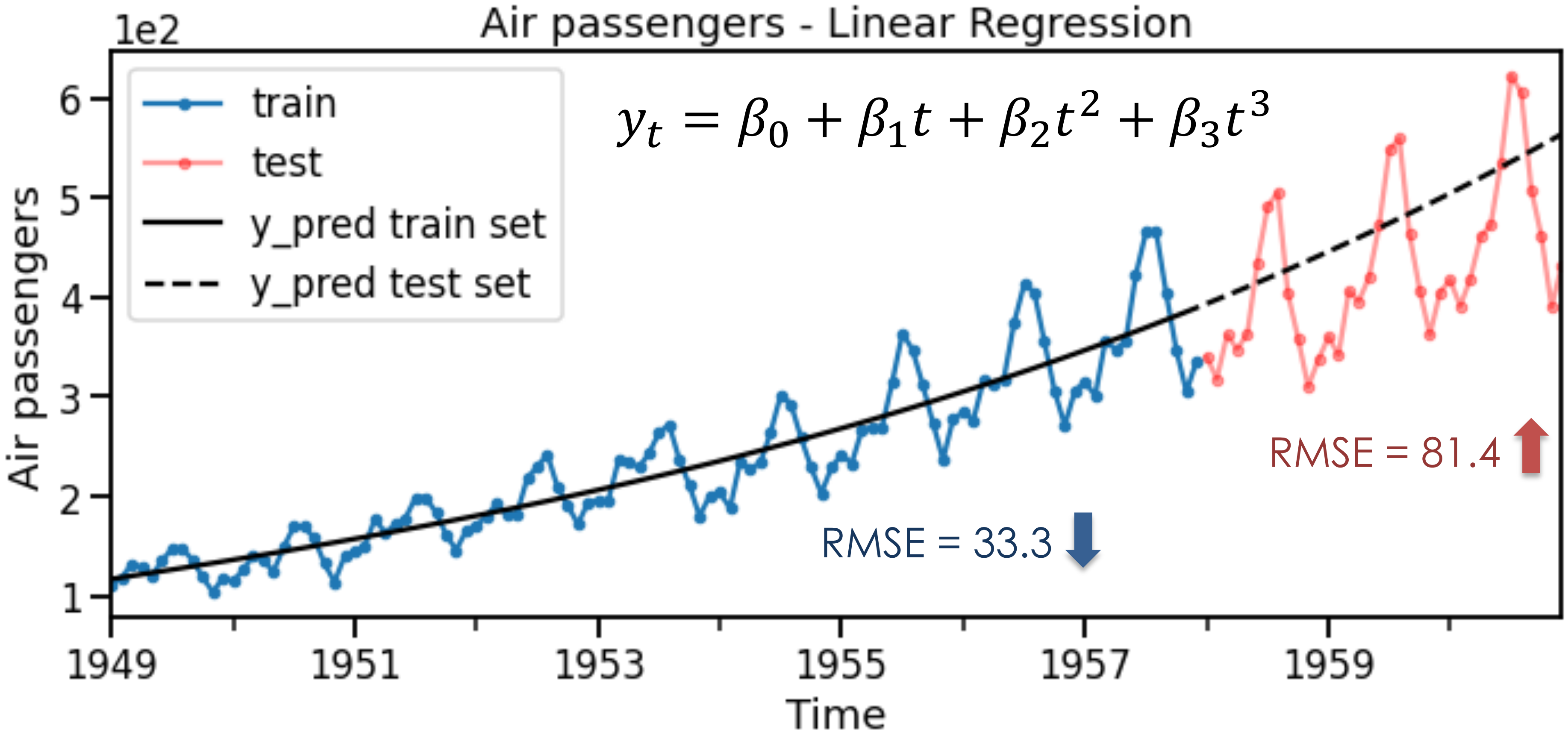
Example: Linear regression with t



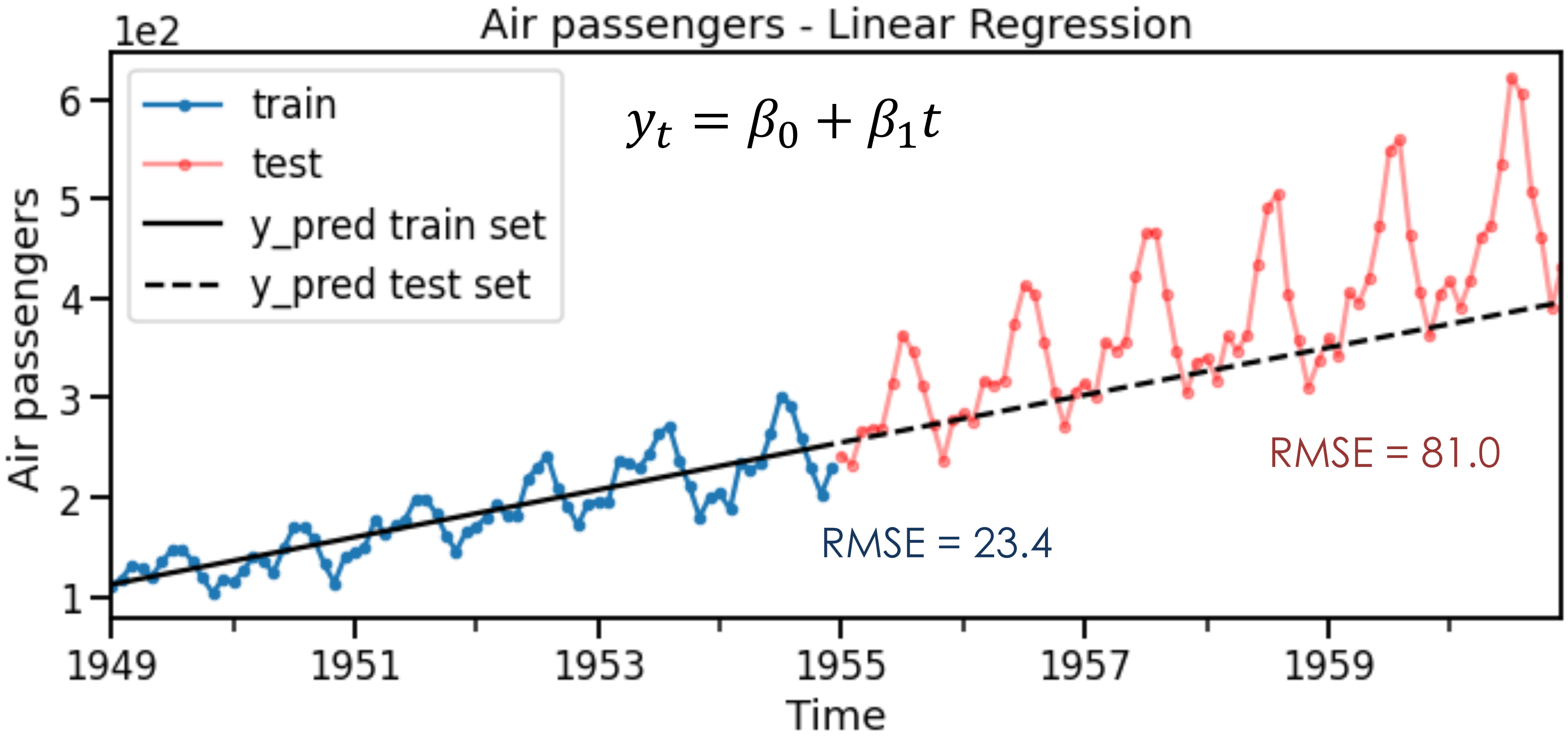
Example: Linear regression with t and t^2



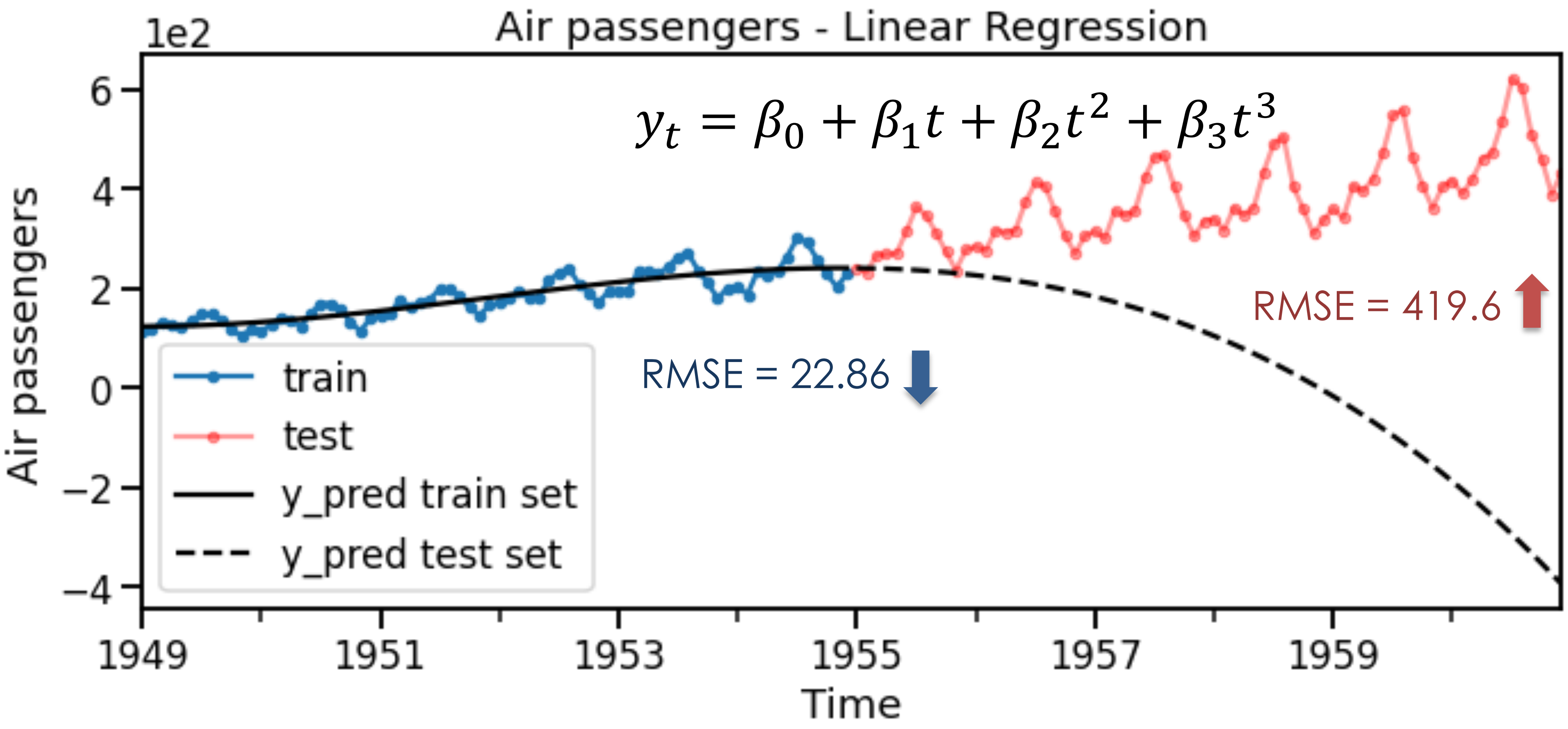
Example: Linear regression with t, t^2, t^3



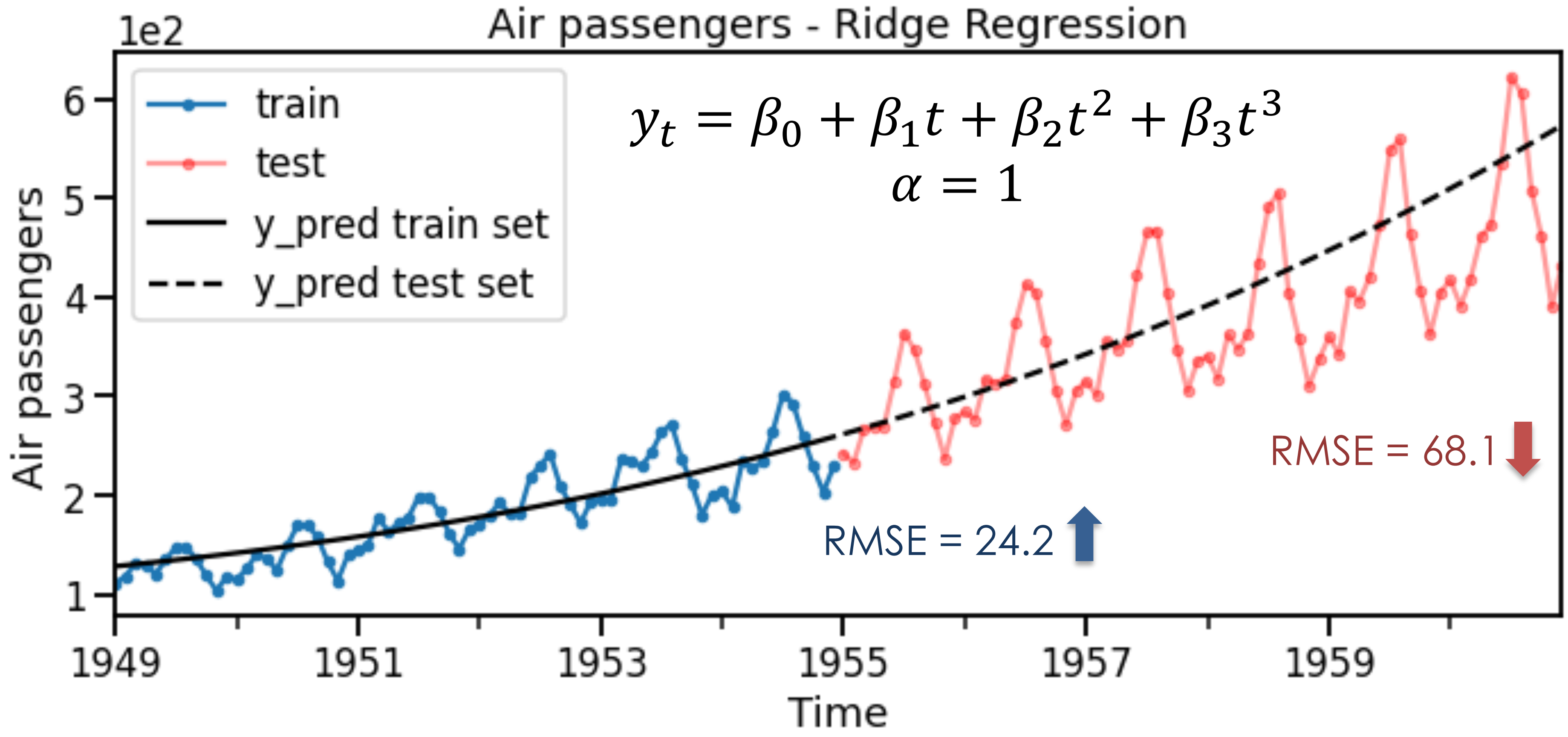
Example: Linear regression with t & less data



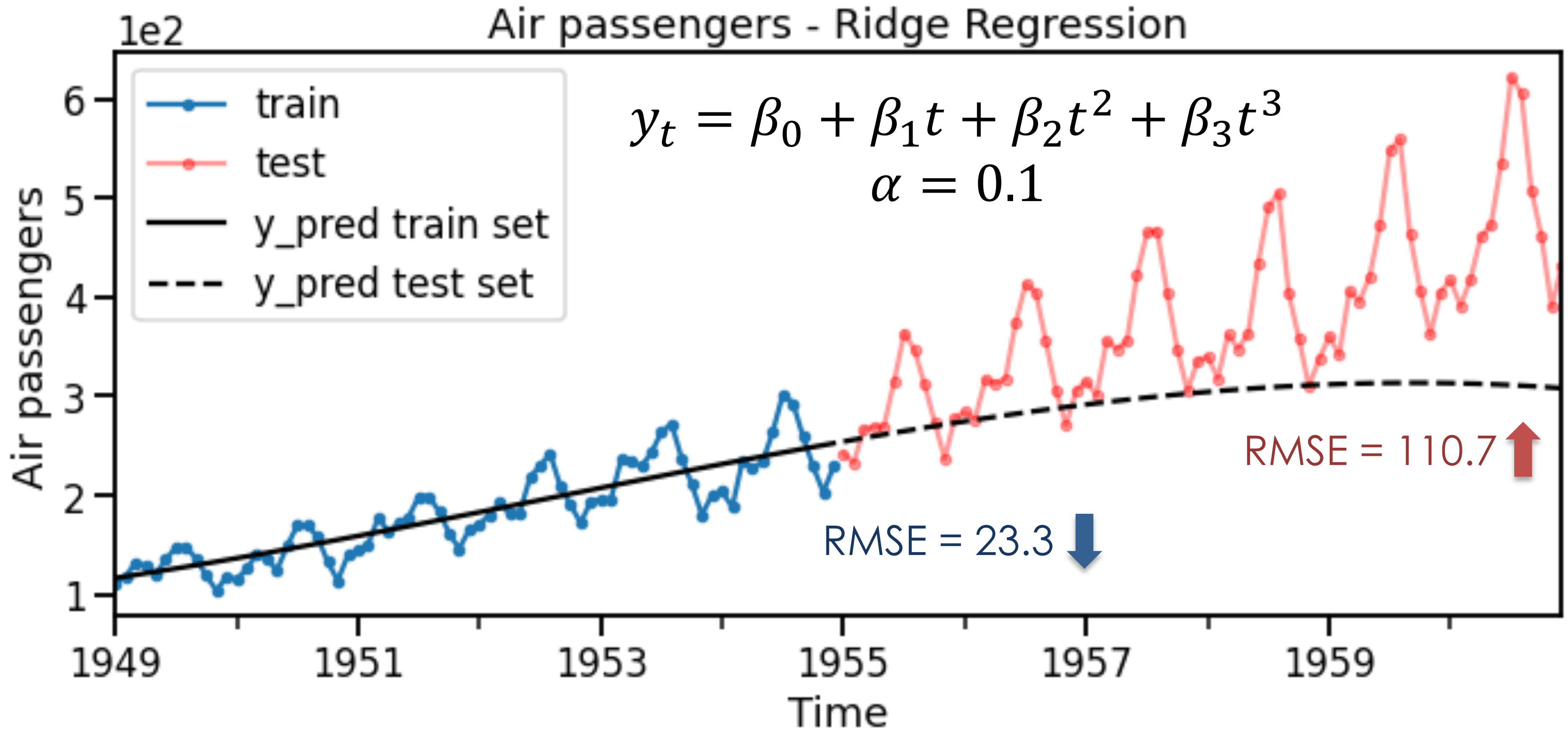
Example: Linear regression with t, t^2, t^3 & less data



Example: Ridge regression with t, t^2, t^3 & less data



Example: Ridge regression with t, t^2, t^3 & less data



Implementation - Pandas

```
df["time_since_1949-01_2"] = df["time_since_1949-01"]**2
df.head()
```

ds	y	time_since_1949-01	time_since_1949-01_2
1949-01	112	0	0
1949-02	118	1	1
1949-03	132	2	4
1949-04	129	3	9
1949-05	121	4	16

Implementation

sklearn.preprocessing.PolynomialFeatures

class `sklearn.preprocessing.PolynomialFeatures`(*degree=2*, ***, *interaction_only=False*, *include_bias=True*, *order='C'*)

[source]

Generate polynomial and interaction features.

Generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2].

Read more in the User Guide.

Implementation

```
# Create and use the polynomial transformer.
poly_transformer = PolynomialFeatures(degree=2, # degree of polynomial
                                      include_bias=False # exclude constant term
                                     )

# Create polynomial features from a given column
result = poly_transformer.fit_transform(df[["time_since_1949-01"]])
result
```

Implementation

```python
# Create and use the polynomial transformer.
poly_transformer = PolynomialFeatures(degree=2, # degree of polynomial
                                      include_bias=False # exclude constant term
                                     )

# Create polynomial features from a given column
result = poly_transformer.fit_transform(df[["time_since_1949-01"]])
result
```

	time_since_1949-01	time_since_1949-01^2
ds		
1949-01	0.0	0.0
1949-02	1.0	1.0
1949-03	2.0	4.0
1949-04	3.0	9.0
1949-05	4.0	16.0
...	...	...
	t	t^2

Summary

It is possible but not recommended to use the non-linear time features to model non-linear trends.

There is a risk of overfitting and extrapolating poorly.

Regularisation can help reduce the risk of overfitting.