§2.1 Fitting a linear model

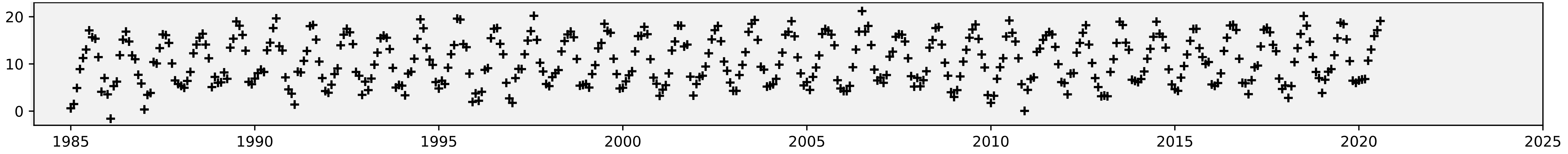
You’ve got to have models in your head. And you’ve got to array your experience – both vicarious and direct – on this latticework of models.

You may have noticed students who just try to remember and pound back what is remembered. Well, they fail in school and in life. You’ve got to hang experience on a latticework of models in your head.

Charlie Munger (business partner of Warren Buffet),
A lesson on elementary, worldly wisdom as it relates to investment management & business.

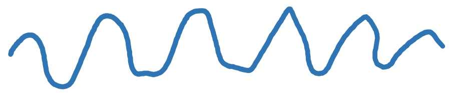
Monthly average temperatures in Cambridge, UK

What's a good model for this dataset?

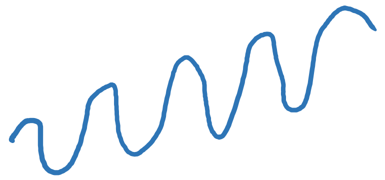


Climate is stable?

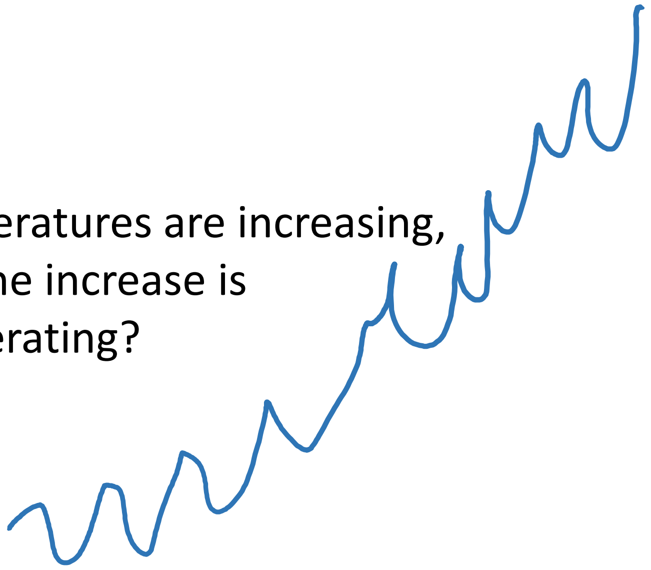
$$\text{Temp}(t) \sim a + b\sin\big(2\pi(t+\phi)\big) + N(0,\sigma^2)$$



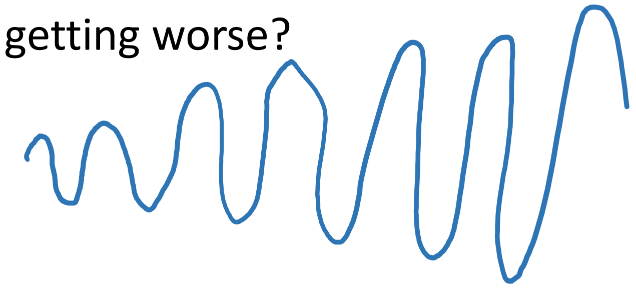
Temperatures are increasing?



Temperatures are increasing, and the increase is accelerating?



The extremes are getting worse?



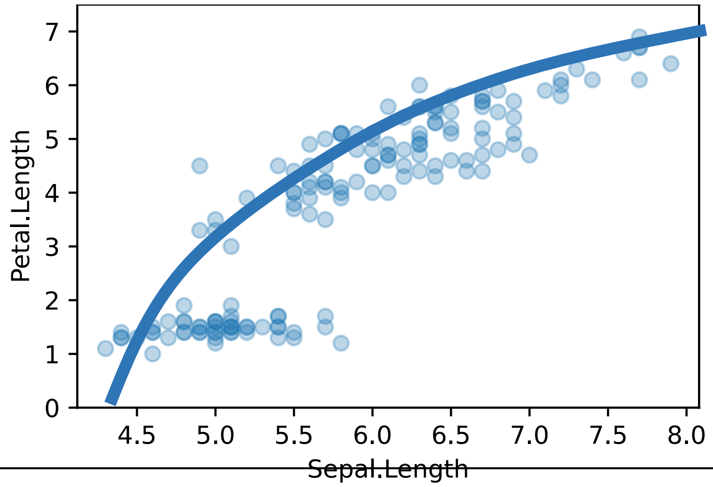
There are so many possible models. We want to make it easy to invent and fit new models, so we have time to explore all the possibilities.

Example 2.1.1

The Iris dataset has 50 records of iris measurements, from three species.

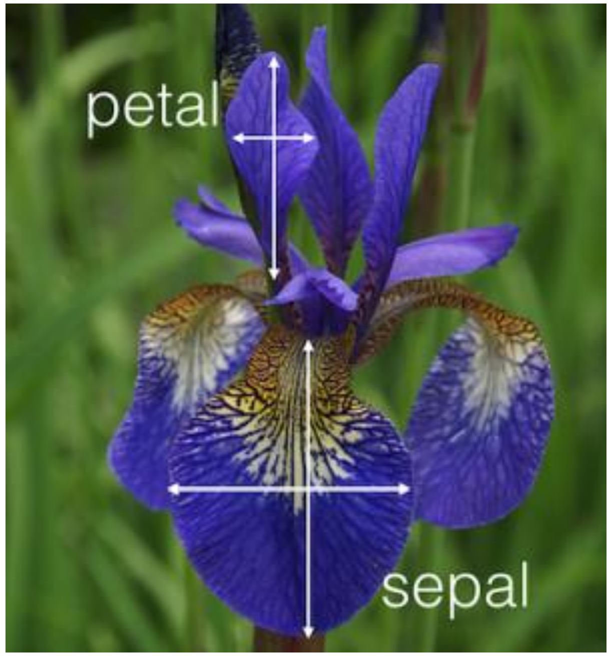
Petal. Length	Petal. Width	Sepal. Length	Sepal. Width	Species
1.0	0.2	4.6	3.6	setosa
5.0	1.9	6.3	2.5	virginica
5.8	1.6	7.2	3.0	virginica
4.2	1.2	5.7	3.0	versicolor
…				

How does `Petal.Length` depend on `Sepal.Length`?



Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$

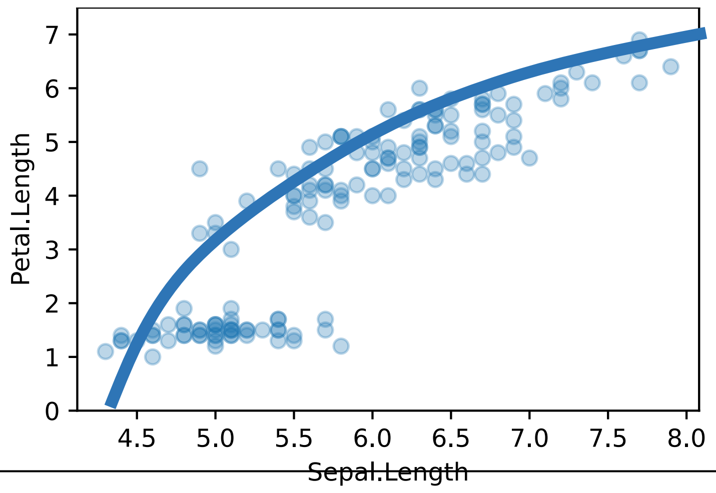


Example 2.1.1

The Iris dataset has 50 records of iris measurements, from three species.

Petal. Length	Petal. Width	Sepal. Length	Sepal. Width	Species
1.0	0.2	4.6	3.6	setosa
5.0	1.9	6.3	2.5	virginica
5.8	1.6	7.2	3.0	virginica
4.2	1.2	5.7	3.0	versicolor
…				

How does `Petal.Length` depend on `Sepal.Length`?



Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$

Linear Model

unknown parameters to be estimated

$$\begin{bmatrix} PL_1 \\ PL_2 \\ \vdots \\ PL_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} SL_1 \\ SL_2 \\ \vdots \\ SL_n \end{bmatrix} + \gamma \begin{bmatrix} (SL_1)^2 \\ (SL_2)^2 \\ \vdots \\ (SL_n)^2 \end{bmatrix}$$

response vector

feature vectors

Not a linear model

$$Temp \approx \alpha + \beta \sin\left(2\pi(t+\phi)\right) + \gamma t$$

Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),
$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$

Supervised learning

Response & features are numeric

Response is predicted by a linear combination of (known) feature vectors, weighted by unknown parameters.

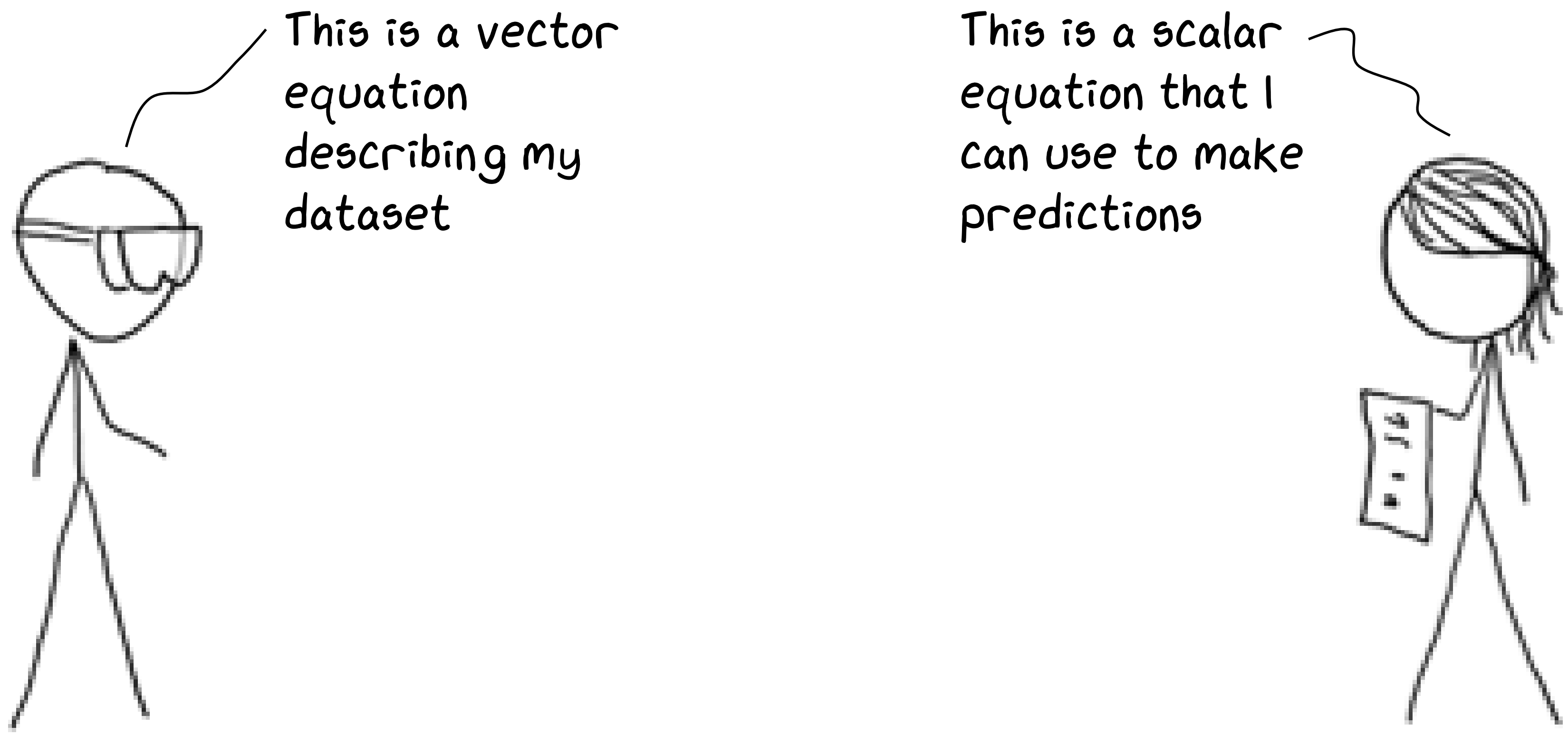
$$\begin{bmatrix} \mathrm{PL}_1 \\ \mathrm{PL}_2 \\ \vdots \\ \mathrm{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \mathrm{SL}_1 \\ \mathrm{SL}_2 \\ \vdots \\ \mathrm{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\mathrm{SL}_1)^2 \\ (\mathrm{SL}_2)^2 \\ \vdots \\ (\mathrm{SL}_n)^2 \end{bmatrix}$$

Models of this form are called *linear models* (because they're based on linear algebra).

They are flexible, and very fast to optimize.

NOTATION WARNING

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$


This is a vector equation describing my dataset
This is a scalar equation that I can use to make predictions

Least squares estimation

Consider a linear model

$$y \approx \beta_1 e_1 + \cdots + \beta_K e_K$$

“All models are wrong.”

The vector of prediction errors is called the *residual vector*,

$$\varepsilon = y - (\beta_1 e_1 + \cdots + \beta_K e_K)$$

We can fit the model using *least squares estimation.* This means finding parameters $\beta_1, \ldots, \beta_K$ to minimize the mean square error

$$\text{mse} = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i^2$$

```
iris = pandas.read_csv(...)
```

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma (\texttt{Sepal.Length})^2$$

$$\begin{bmatrix} \mathrm{PL}_1 \\ \mathrm{PL}_2 \\ \vdots \\ \mathrm{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \mathrm{SL}_1 \\ \mathrm{SL}_2 \\ \vdots \\ \mathrm{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\mathrm{SL}_1)^2 \\ (\mathrm{SL}_2)^2 \\ \vdots \\ (\mathrm{SL}_n)^2 \end{bmatrix}$$

Fitting the model

```
one, SL, PL = np.ones(len(iris)), iris['Sepal.Length'], iris['Petal.Length']
model = sklearn.linear_model.LinearRegression(fit_intercept=False)
model.fit(np.column_stack([one, SL, SL**2]), PL)
(α,β,γ) = model.coef_
```

Making predictions / getting fitted values from the model

```
newSL = np.linspace(4.2, 8.2, 20)
predPL = α + β*newSL + γ*(newSL**2)
```

```
iris = pandas.read_csv(...)
```

$$\text{Petal.Length} \approx \alpha + \beta\ \text{Sepal.Length} + \gamma (\text{Sepal.Length})^2$$

$$\begin{bmatrix} PL_1 \\ PL_2 \\ \vdots \\ PL_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} SL_1 \\ SL_2 \\ \vdots \\ SL_n \end{bmatrix} + \gamma \begin{bmatrix} (SL_1)^2 \\ (SL_2)^2 \\ \vdots \\ (SL_n)^2 \end{bmatrix}$$

sklearn.linear_model puts in the $[\,\vdots\,]$ feature for us by default. If we don't want it we have to specify fit_intercept=False.

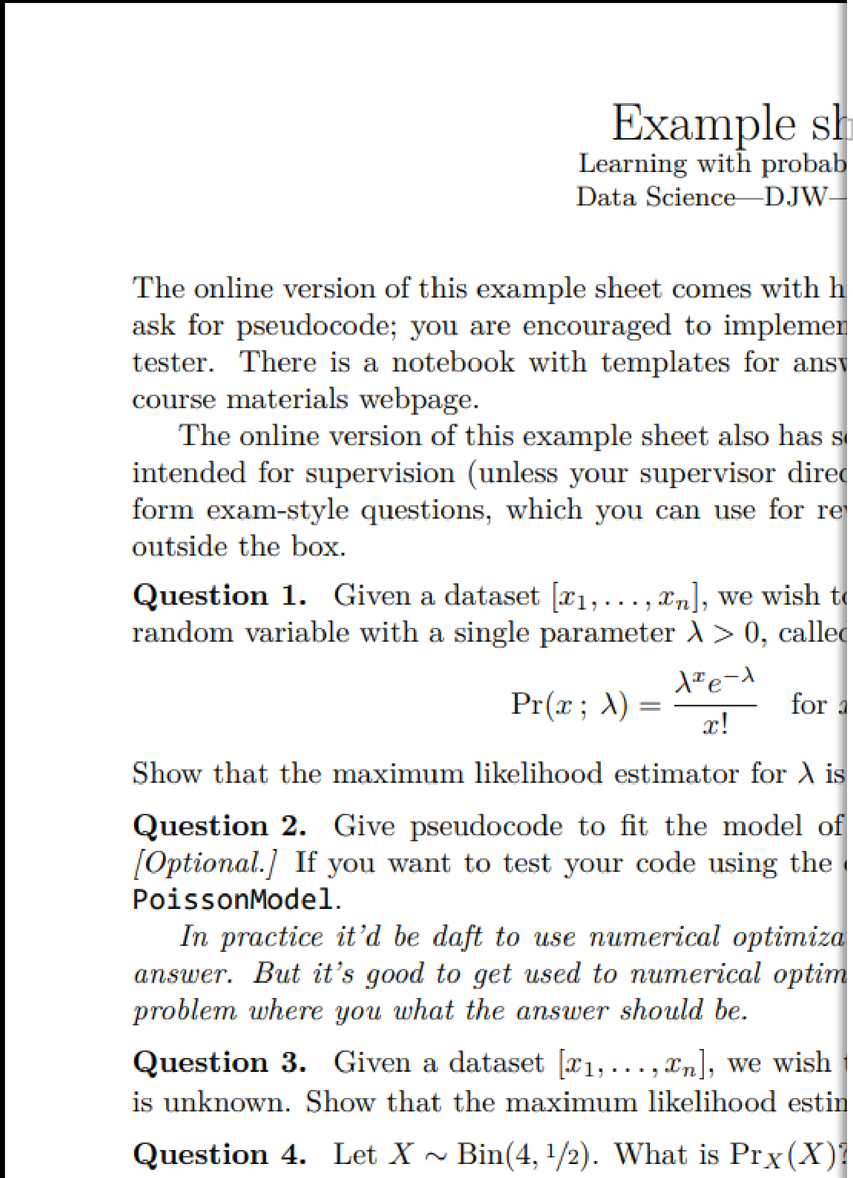
Fitting the model (cleaner code)

```
SL, PL = iris['Sepal.Length'], iris['Petal.Length']
model = sklearn.linear_model.LinearRegression()
model.fit(np.column_stack([SL, SL**2]), PL)
α,(β,γ) = model2.intercept_, model2.coef_
```

This saves us from having to explicitly code up the prediction formula — so there's less chance we introduce bugs.

Making predictions / getting fitted values from the model (cleaner code)

```
newSL = np.linspace(4.2, 8.2, 20)
predPL = model.predict(np.column_stack([newSL, newSL**2]))
```



Example sh

Learning with probab

Data Science—DJW—

The online version of this example sheet comes with h
ask for pseudocode; you are encouraged to implemen
tester. There is a notebook with templates for ansv
course materials webpage.

The online version of this example sheet also has s
intended for supervision (unless your supervisor dire
form exam-style questions, which you can use for re
outside the box.

Question 1. Given a dataset $[x_1, \dots, x_n]$, we wish t
random variable with a single parameter $\lambda > 0$, called

$$\Pr(x\,;\,\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \quad \text{for } x$$

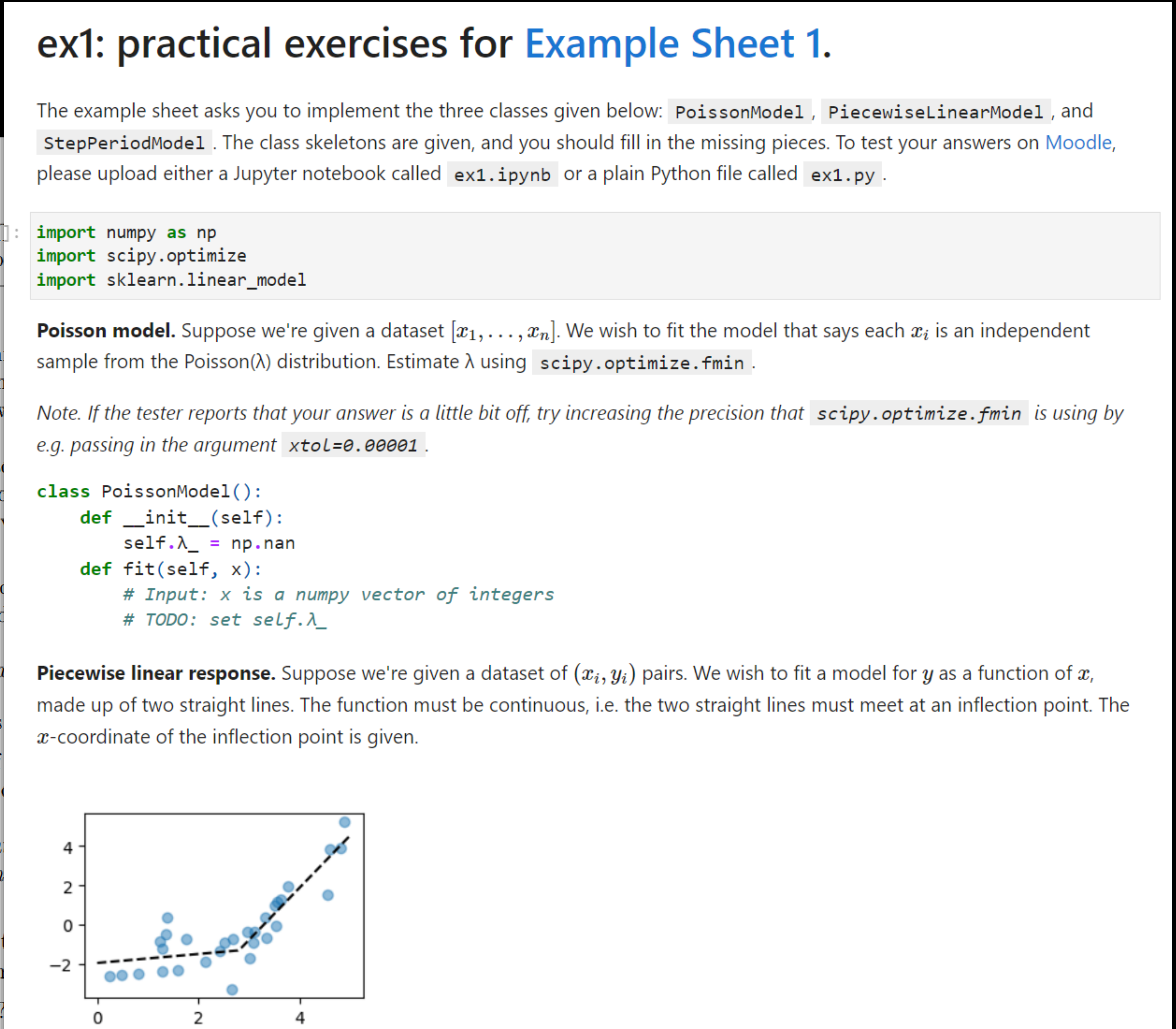
Show that the maximum likelihood estimator for λ is

Question 2. Give pseudocode to fit the model of
[Optional.] If you want to test your code using the
`PoissonModel`.

*In practice it'd be daft to use numerical optimiza
answer. But it's good to get used to numerical optim
problem where you what the answer should be.*

Question 3. Given a dataset $[x_1, \dots, x_n]$, we wish
is unknown. Show that the maximum likelihood estin

Question 4. Let $X \sim \text{Bin}(4, 1/2)$. What is $\Pr_X(X)$



ex1: practical exercises for Example Sheet 1.

The example sheet asks you to implement the three classes given below: `PoissonModel`, `PiecewiseLinearModel`, and `StepPeriodModel`. The class skeletons are given, and you should fill in the missing pieces. To test your answers on Moodle, please upload either a Jupyter notebook called `ex1.ipynb` or a plain Python file called `ex1.py`.

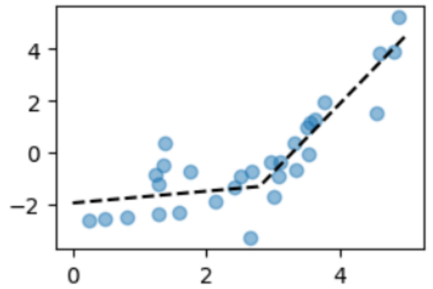
```
import numpy as np
import scipy.optimize
import sklearn.linear_model
```

Poisson model. Suppose we're given a dataset $[x_1, \dots, x_n]$. We wish to fit the model that says each x_i is an independent sample from the Poisson(λ) distribution. Estimate λ using `scipy.optimize.fmin`.

Note. If the tester reports that your answer is a little bit off, try increasing the precision that `scipy.optimize.fmin` is using by e.g. passing in the argument `xtol=0.00001`.

```
class PoissonModel():
    def __init__(self):
        self.λ_ = np.nan
    def fit(self, x):
        # Input: x is a numpy vector of integers
        # TODO: set self.λ_
```

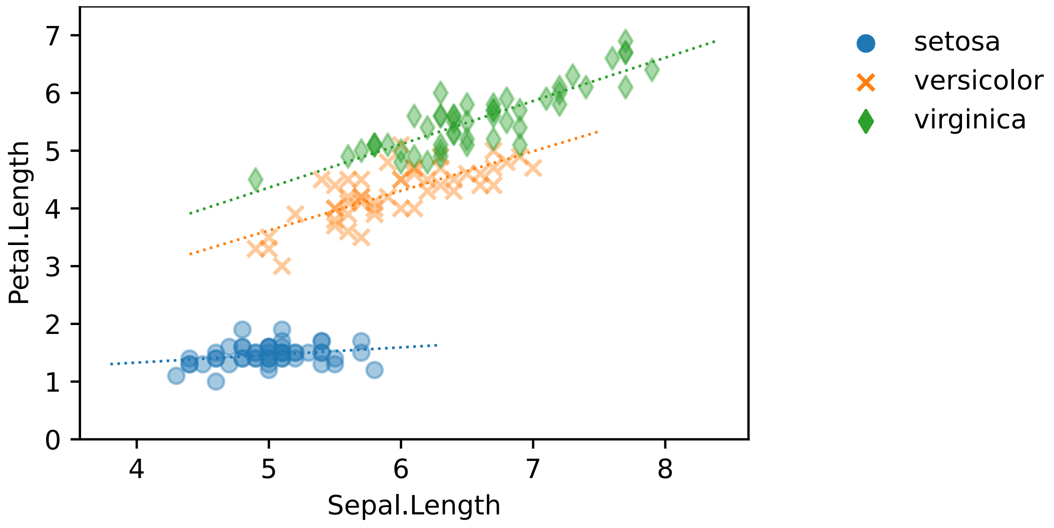
Piecewise linear response. Suppose we're given a dataset of (x_i, y_i) pairs. We wish to fit a model for y as a function of x, made up of two straight lines. The function must be continuous, i.e. the two straight lines must meet at an inflection point. The x-coordinate of the inflection point is given.



§2.2 Feature design

How do we design features, so that linear models answer the questions we want answered?

ONE-HOT CODING



$PL \approx \alpha_{species} + \beta_{species} SL$

This model has 6 unknown parameters:

α_{set} α_{vers} α_{virg}

β_{set} β_{vers} β_{virg}.

So my model has 6 feature vectors.

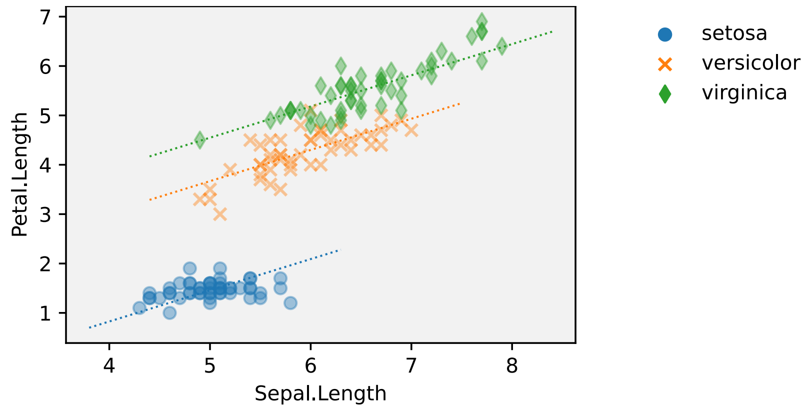
Row 1: setosa : $PL \approx \alpha_{setosa} + \beta_{setosa} SL$.

$$\begin{array}{l} * \text{seto} \\ \text{virg} \\ \text{virg} \\ \text{seto} \\ \text{vers} \\ \\ \end{array} \begin{bmatrix} PL_1 \\ PL_2 \\ PL_3 \\ PL_4 \\ PL_5 \\ \vdots \end{bmatrix} \approx \alpha_{\text{seto}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \alpha_{\text{virg}} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \end{bmatrix} + \alpha_{\text{vers}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \beta_{\text{seto}} \begin{bmatrix} SL_1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \beta_{\text{virg}} \begin{bmatrix} 0 \\ SL_2 \\ SL_3 \\ \vdots \end{bmatrix} + \beta_{\text{vers}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

$$\vec{PL} \approx \alpha_{seto} 1_{\vec{spec}=setosa} + \alpha_{virg} 1_{\vec{spec}=virg} + \alpha_{vers} 1_{\vec{spec}=vers} + \beta_{seto} (\vec{SL} \times 1_{\vec{spec}=seto}) + \cdots$$

EXERCISE

Fit the model with three parallel straight lines.

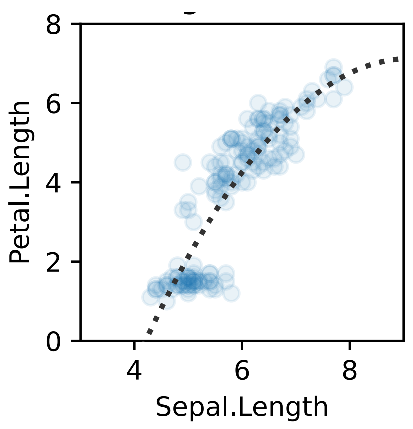


$$PL \approx \alpha_{species} + \beta\, SL$$

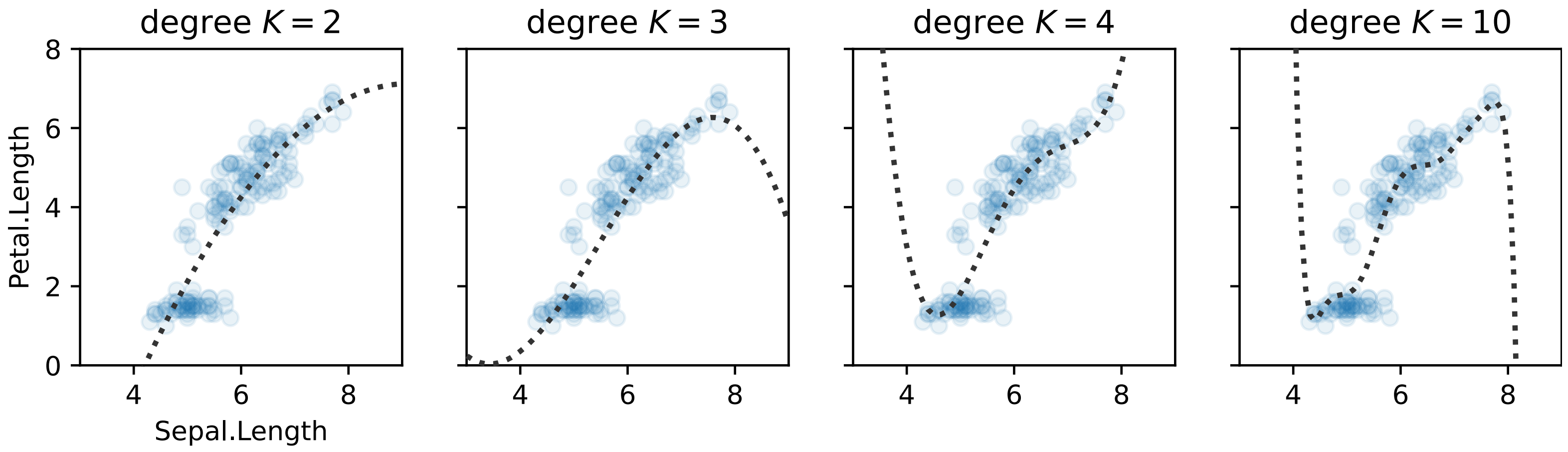
$$\approx \alpha_{set}\, 1_{\vec{spec} = set} + \cdots$$

$$+ \beta\, \vec{SL}$$

NON-LINEAR RESPONSE



$$\text{Petal.Length} \approx \alpha + \beta\ \text{Sepal.Length} + \gamma(\text{Sepal.Length})^2$$



$$\text{Petal.Length} \approx \beta_0 + \sum_{k=1}^{K} \beta_k (\text{Sepal.Length})^k$$

Q. Should we just keep adding more and more features to our model?

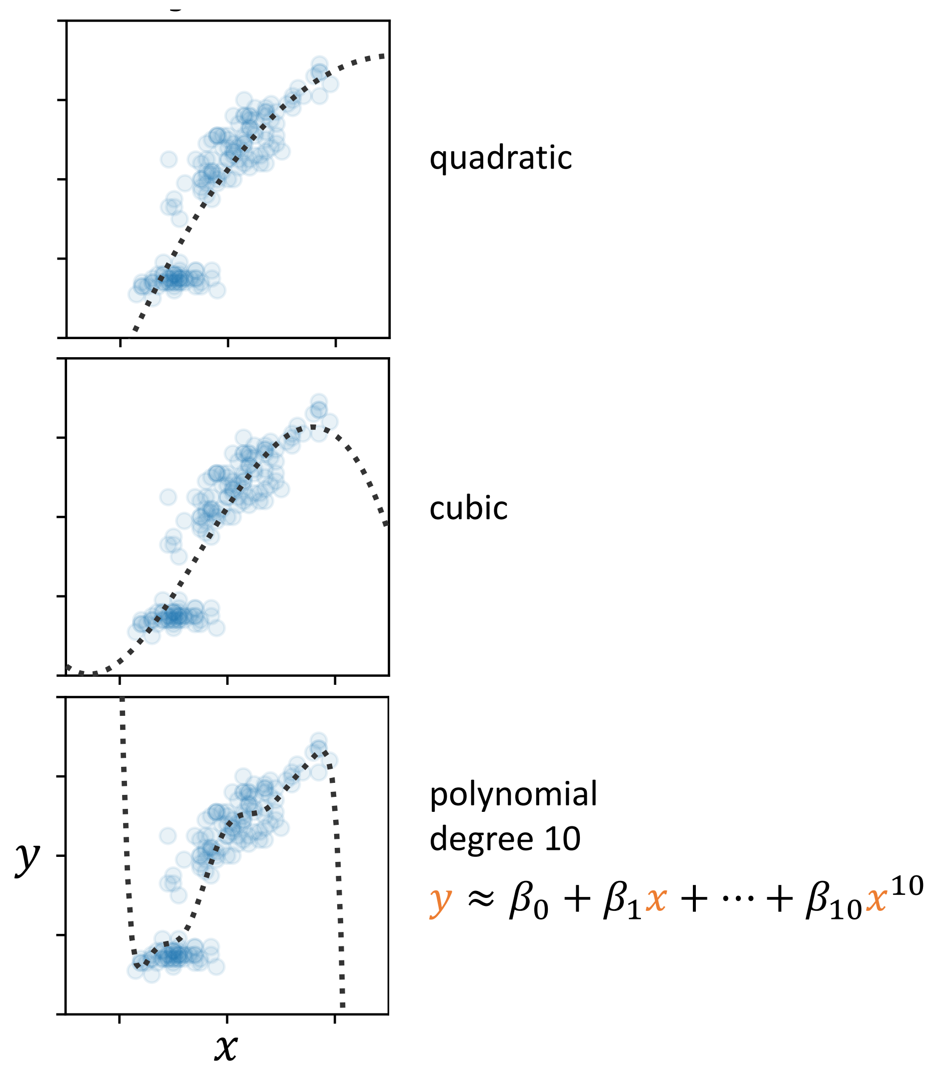
(seeing as the more features we add, the better we can fit the dataset)

A. No. If we did, we'd *overfit*.

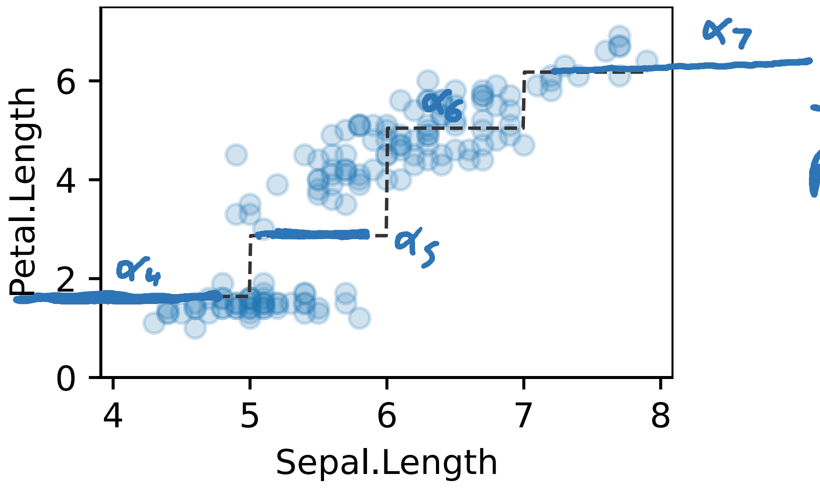
Only add in features that you (as a scientist) believe are relevant.

Or do model testing, e.g. evaluation on a holdout set.

[§5–7]



NON-LINEAR RESPONSE via one-hot coding

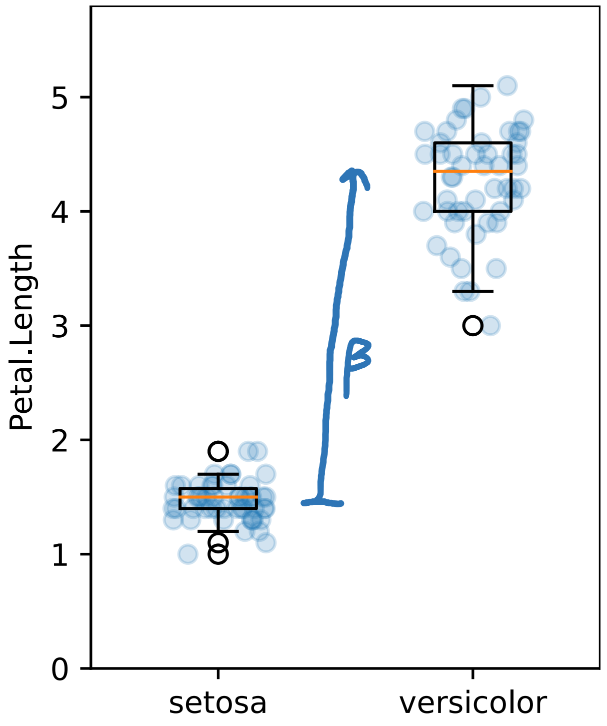


$$\vec{PL} \approx \alpha_4 \, 1_{\vec{SL}<5} + \alpha_5 \, 1_{\lfloor \vec{SL} \rfloor = 5} + \alpha_6 \, 1_{\lfloor \vec{SL} \rfloor = 6} + \alpha_7 \, 1_{\vec{SL} \geq 7}$$

e.g. for an observation with $SL = 5.3$ we predict α_5

$SL = 2.7$ α_4

COMPARING GROUPS



Measurements for condition A: a = [$a_1, a_2, \dots, a_m$]
Measurements for condition B: b = [$b_1, b_2, \dots, b_n$]

Can we use a linear model to compare A and B?

cond	x
A	a_1
A	$\vdots$
$\vdots$	
A	a_m
B	b_1
B	b_2
$\vdots$	$\vdots$
B	b_n

$$\vec{x} \approx \alpha_A 1_{\vec{cond}=A} + \alpha_B 1_{\vec{cond}=B}$$

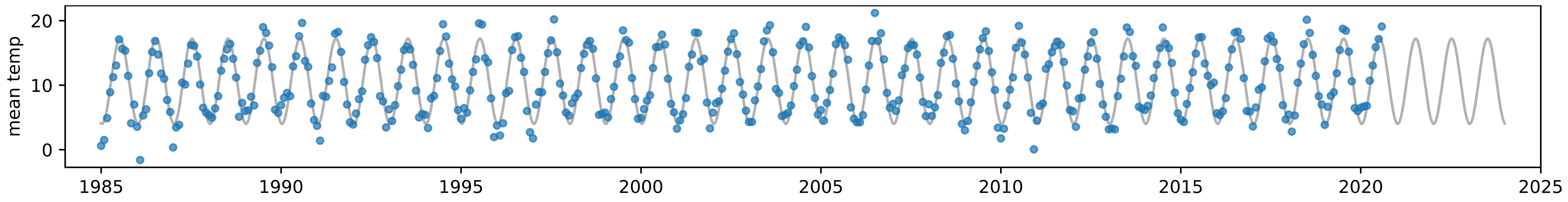
Or

$$\vec{x} = \alpha + \beta 1_{\vec{cond}=B}.$$

For a person of type A, $x \approx \alpha$
B, $x \approx \alpha + \beta$

β measures the difference between the two groups.

PERIODIC PATTERN



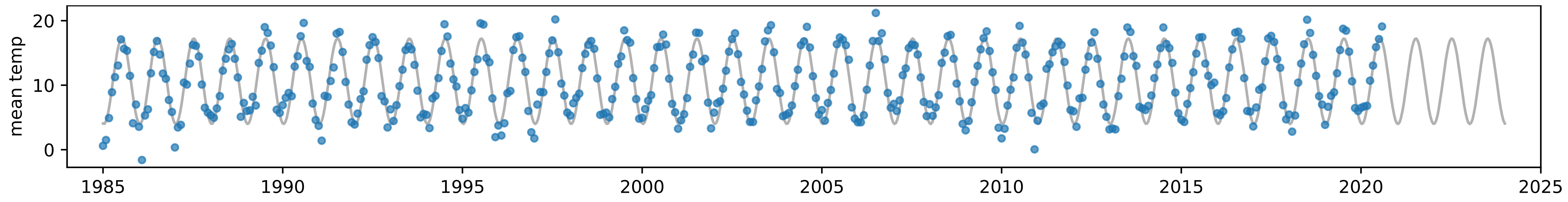
We'd like to fit the model:

$$\texttt{temp} \approx \alpha + \beta \sin(2\pi \texttt{t} + \varphi)$$

It looks like we can't use `sklearn.LinearRegression`. That's only for linear models, e.g.

$$\texttt{temp} \approx \alpha + \beta \texttt{e} + \gamma \texttt{f}$$

PERIODIC PATTERN



We'd like to fit the model:

$$\texttt{temp} \approx \alpha + \beta \sin(2\pi \texttt{t} + \varphi)$$

$$\approx \alpha + \beta \{\sin(2\pi \texttt{t}) \cos\phi \ + \ \cos(2\pi \texttt{t}) \sin\phi\}$$

$$= \alpha + \underbrace{(\beta\cos\phi)}_{} \sin(2\pi \texttt{t}) + \underbrace{(\beta\sin\phi)}_{} \cos(2\pi \texttt{t})$$

$$= \alpha \ + \ \beta_1 \underbrace{\sin(2\pi \texttt{t})}_{} \ + \ \beta_2 \underbrace{\cos(2\pi \texttt{t})}_{}$$

$$= \alpha \ + \ \beta_1 e \ + \ \beta_2 f$$

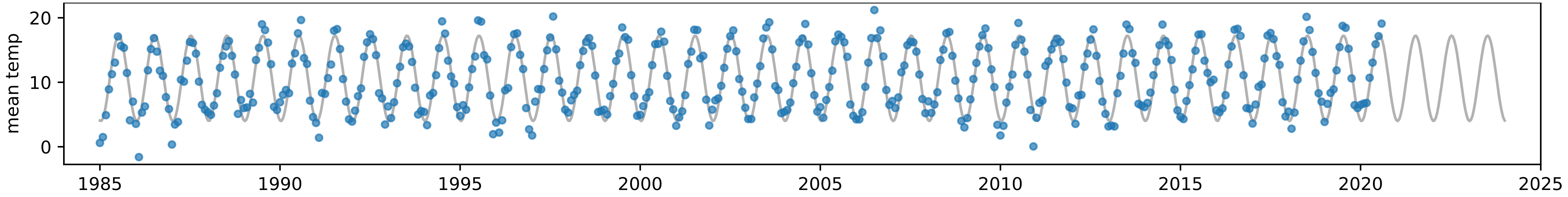
From school trigonometry,

$$\sin(A+B) = \sin(A)\cos(B) + \cos(A)\sin(B)$$

a linear model with feature vectors $\texttt{1}$, $\sin(2\pi\texttt{t})$, $\cos(2\pi\texttt{t})$

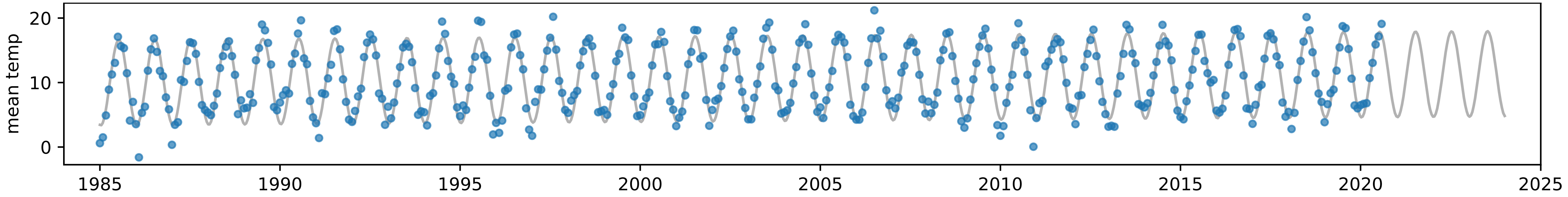
PERIODIC PATTERN

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t})$$



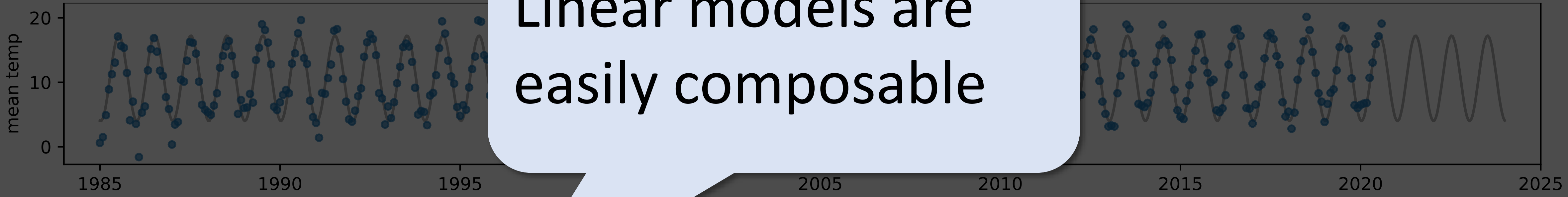
PERIODIC PATTERN + SECULAR TREND

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t}) + \gamma \texttt{t}$$



PERIODIC PATTERN

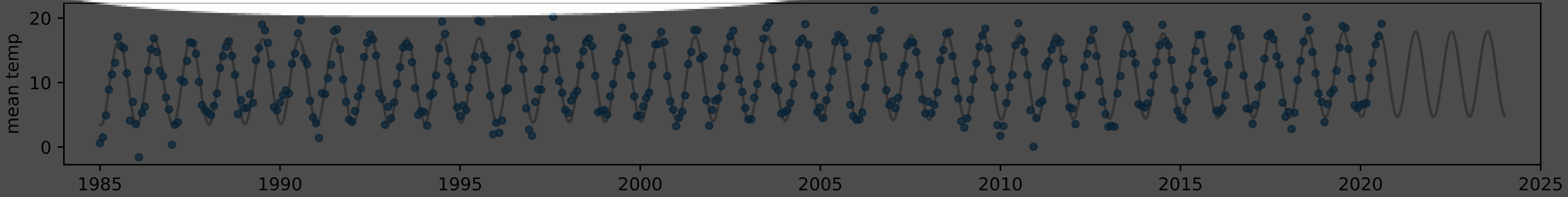
$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta$



Linear models are easily composable

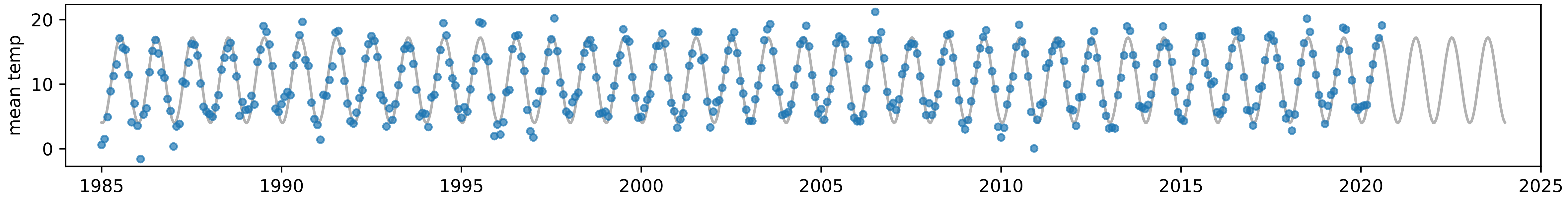
PERIODIC PATTERN + SECULAR TREND

$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t}) + \gamma \texttt{t}$



With our periodic model …

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t})$$

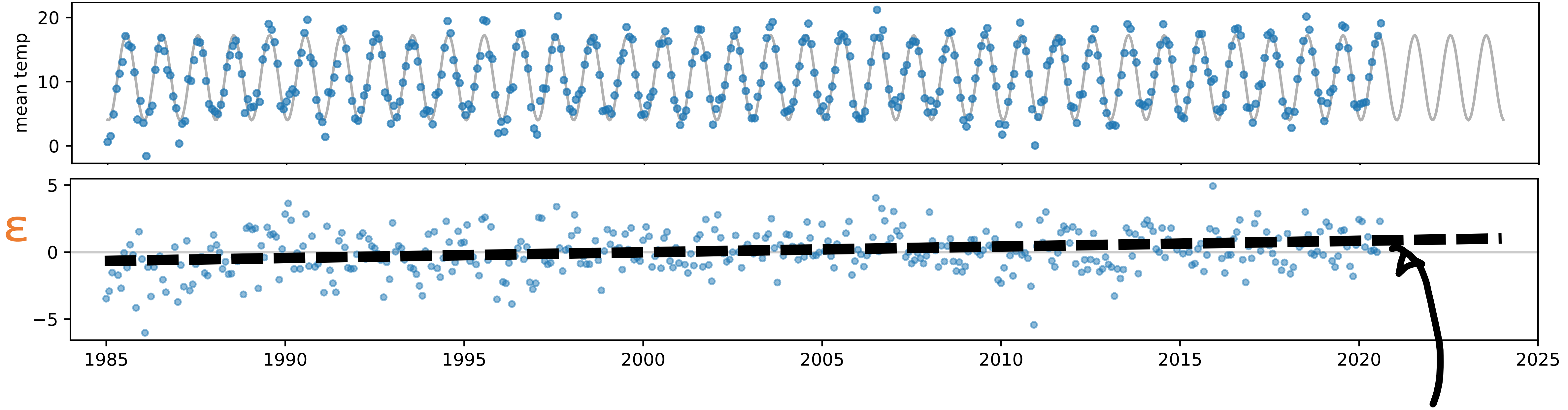


… how do we *discover* we should add a secular term $+\gamma \texttt{t}$?

If we hadn't thought to include climate change in our temperature model ...

$$\texttt{temp} \approx \alpha + \beta \sin\big(2\pi(\texttt{t} + \phi)\big)$$

$$\texttt{temp} = \alpha + \beta \sin\big(2\pi(\texttt{t} + \phi)\big) + \varepsilon$$



$\varepsilon \approx \delta + \gamma t$

This suggests a revised model ...

$$\texttt{temp} = \alpha' + \beta' \sin\big(2\pi(\texttt{t} + \phi)\big) + \gamma \texttt{t} + \varepsilon$$

Diagnosing a linear model

After fitting a model

```
    model.fit(..., y)
```

1. Compute the residuals

```
    ε = y – model.predict()
```

2. Plot ε every way we can think of. If there's a systematic pattern, add a feature that describes it.