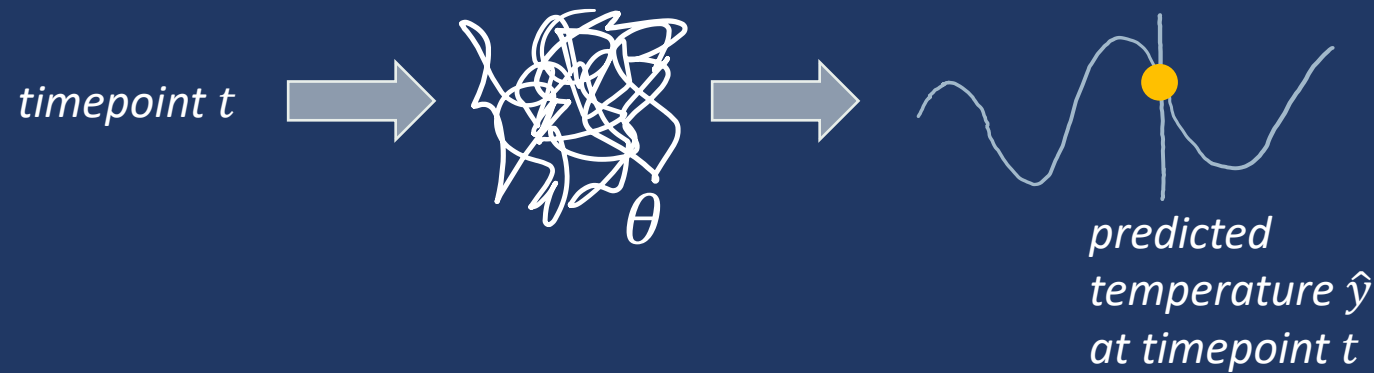


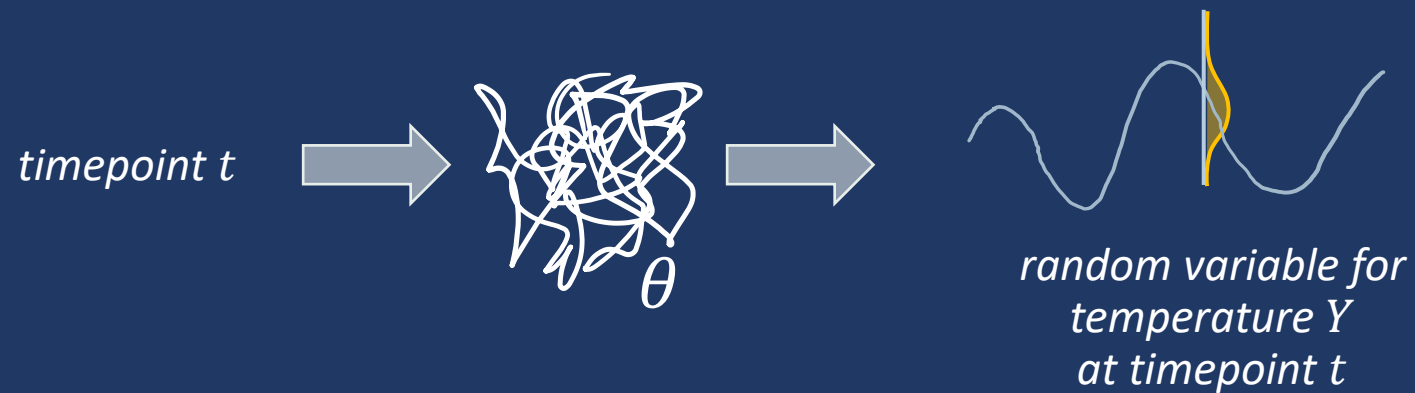
Two different views of regression-type ML

PREDICTION MINDSET



Goal:
minimize prediction error

PROBABILITY MODELLER'S VIEW



Goal:
fit my probability model, using
maximum likelihood estimation

Two different views of classification-type ML

PREDICTION MINDSET



input image



5
*predicted
label $\hat{y}$*

Goal:

maximize prediction accuracy 🤔

but we have to fudge the loss function
to make it differentiable 😬

PROBABILITY MODELLER'S VIEW



input image



5 with prob 45%
3 with prob 41%
6 with prob 5%
...

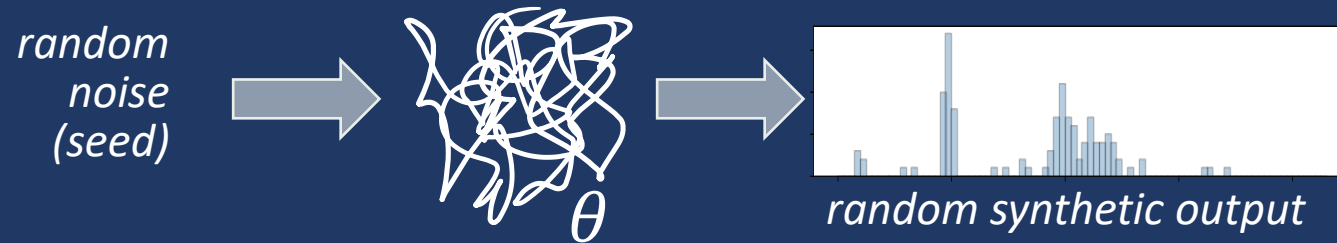
random predicted label Y

Goal:

fit my probability model, using
maximum likelihood estimation

Two different views of generative-type ML

PREDICTION MINDSET

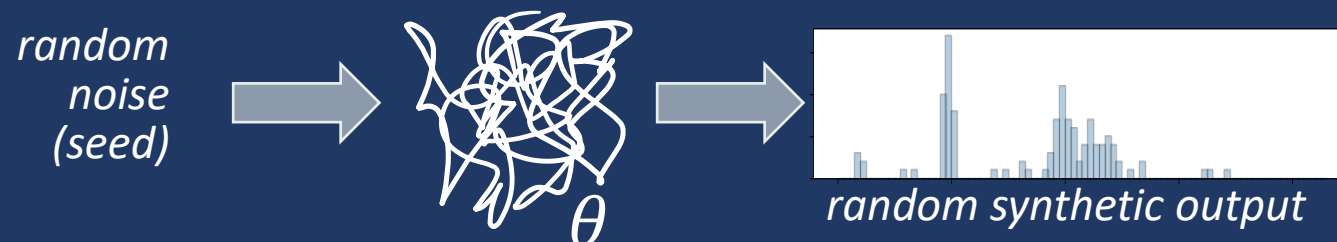


Goal:

Prediction accuracy?

Wait, what? There isn't even any prediction going on here! 🙄

PROBABILITY MODELLER'S VIEW



Goal:

fit my probability model, using maximum likelihood estimation

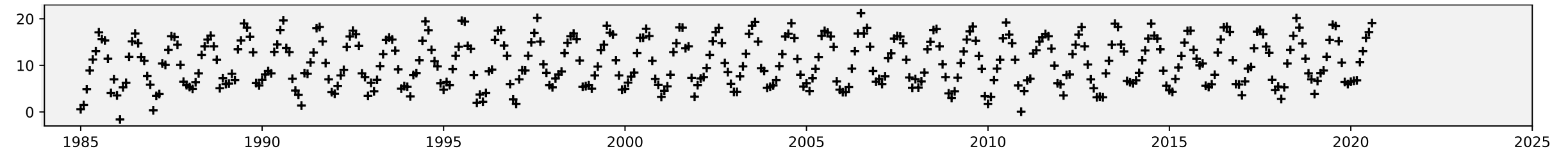
§2.1

Fitting a linear model

(This will look like prediction-mindset thinking, for now. Stay tuned until next Monday.)

Monthly average temperatures in Cambridge, UK

What's a good model for this dataset?

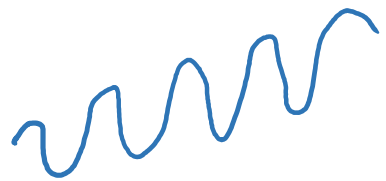


Climate is stable?

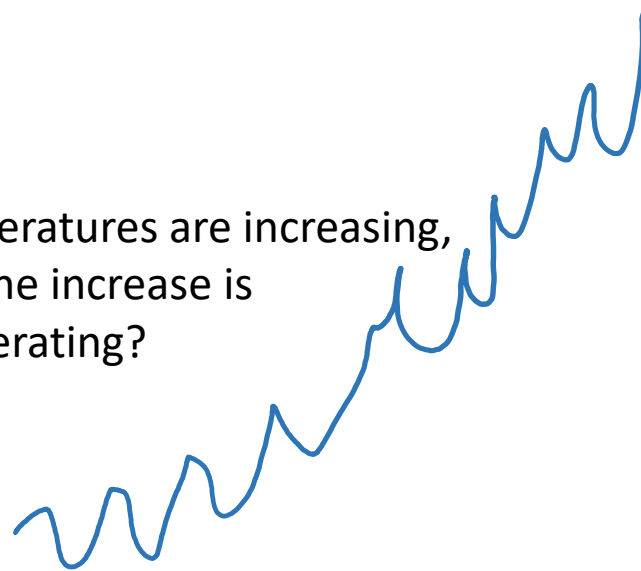
$$\text{Temp}(t) \sim a + b \sin(2\pi(t + \phi)) + N(0, \sigma^2)$$



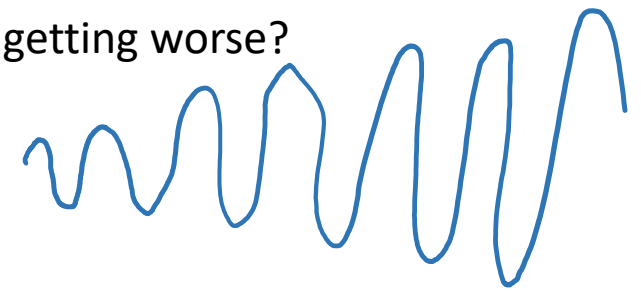
Temperatures are increasing?



Temperatures are increasing,
and the increase is
accelerating?



The extremes are
getting worse?



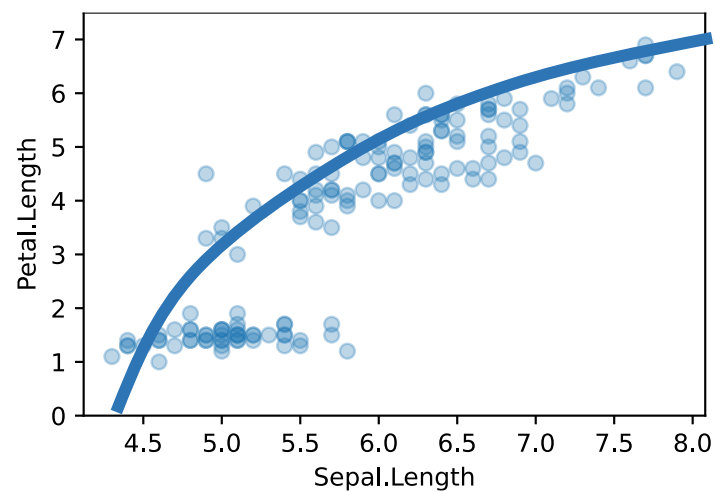
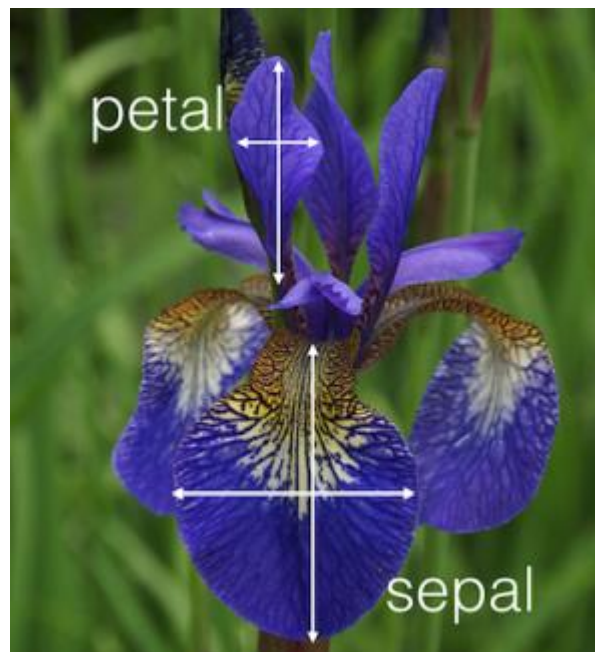
There are so many possible models. We want to make it easy to invent and fit new models, so we have time to explore all the possibilities.

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),
Petal.Length $\approx \alpha + \beta$ **Sepal.Length** + γ (**Sepal.Length**)²

Ask Gemini

7

kaggle.com/competitions/titanic

Install

+

Search

Login

Sign Up

Kaggle · Getting Started Prediction Competition · Ongoing

Join Competition

Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

OverviewDataCodeModelsDiscussionLeaderboardRules

Overview

This competition runs indefinitely with a rolling leaderboard. [Learn more](#)

Description

👋🏡 Ahoy, welcome to Kaggle! You're in the right place.

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

If you want to talk with other users about this competition, come join our Discord! We've got channels for competitions, job postings and career discussions, resources, and socializing with your fellow data scientists. Follow

Competition Host

Kaggle

Prizes & Awards

Knowledge

Does not award Points or Medals

Participation

1,477,000 Entrants

11,329 Participants

11,270 Teams

41,524 Submissions

Tags

Binary Classification

Tabular

Beginner

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

Applying this to each row of the dataset, we get n approximations:

$$\begin{array}{ll} \text{Row 1 of dataset:} & \text{Petal.Length}_1 \approx \alpha + \beta \text{ Sepal.Length}_1 + \gamma (\text{Sepal.Length}_1)^2 \\ \text{Row 2 of dataset:} & \text{Petal.Length}_2 \approx \alpha + \beta \text{ Sepal.Length}_2 + \gamma (\text{Sepal.Length}_2)^2 \\ & \vdots \\ \text{Row } n \text{ of dataset:} & \text{Petal.Length}_n \approx \alpha + \beta \text{ Sepal.Length}_n + \gamma (\text{Sepal.Length}_n)^2 \end{array}$$

It's convenient to write it in vector form:

unknown parameters
to be estimated

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

response
vector feature
 vectors

A linear model is

- ❖ a prediction model
[it has a response variable, and feature variables]
- ❖ in which the response and the features are numeric
- ❖ and the response vector is predicted by a linear combination of (known) feature vectors weighted by (unknown) parameters

They're called *linear models* because they're based on linear algebra (i.e. vectors).

They are flexible, and very fast to optimize.

A linear model:

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

Not a linear model:



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

Why is this not a linear model?

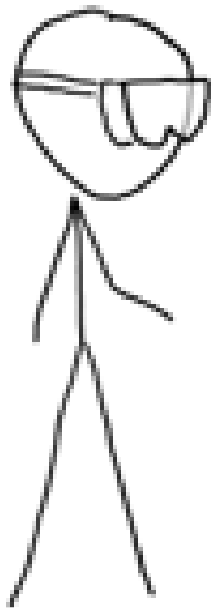
Because of ϕ inside $\sin$

Note: the nonlinear function $\sin$ isn't intrinsically a problem. Here's a model that includes $\sin$ but is linear:

$$\text{temp} \approx \alpha + \beta \sin(2\pi t) + \gamma t$$



$$\text{Petal.Length} \approx \alpha \vec{1} + \beta \text{Sepal.Length} + \gamma (\text{Sepal.Length})^2$$



This is a data scientist's vector equation describing my dataset, one row per datapoint

(And I can add a scalar to a vector, with an implicit vector of ones.)

This is a scientist's modelling equation that applies to scalars, and I can use it to make predictions



How to fit a linear model

Least squares estimation

Consider a linear model

$$y \approx \beta_1 e^{(1)} + \dots + \beta_K e^{(K)}$$

“All models are wrong.”

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

$$\varepsilon = y - (\beta_1 e^{(1)} + \dots + \beta_K e^{(K)})$$

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

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

```

0 import sklearn.linear_model
1 iris = pandas.read_csv(...)

```

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

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

Fitting the model

```

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

```

Making predictions

```

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

```

```

0 import sklearn.linear_model
1 iris = pandas.read_csv(...)

```

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

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

Fitting the model (cleaner code)

```

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

```

sklearn.linear_model puts in the $[1, \dots, 1]$ feature for us by default. If we don't want it, we have to specify `fit_intercept=False`.

DRY way to make predictions

```

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

```

This saves us from having to explicitly code up the prediction formula – less chance we introduce bugs

Example

Learning with p
Data Science—I

Some of the questions ask for pseudocode. I suggest using an online tester. There is a notebook with templates for the course materials webpage.

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

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

Show that the maximum likelihood estimator for λ is given by

Question 2. Give pseudocode to fit the model $y = \lambda x$. [Optional.] If you want to test your code using the `PoissonModel` class,

In practice it'd be daft to use numerical optimization to find the maximum likelihood answer. But it's good to get used to numerical optimization for a problem where you know the answer should be.

Question 3. Given a dataset $[x_1, \dots, x_n]$, we wish to fit a model for a random variable with a single parameter λ that is unknown. Show that the maximum likelihood

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`.

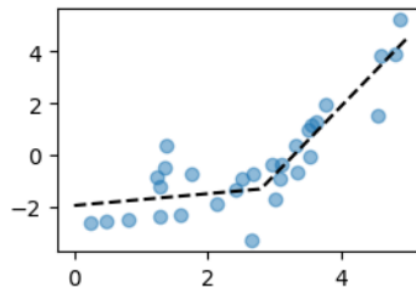
```
import numpy as np
import scipy.optimize
import sklearn.linear_model
```

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 $\text{Poisson}(\lambda)$ 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`.

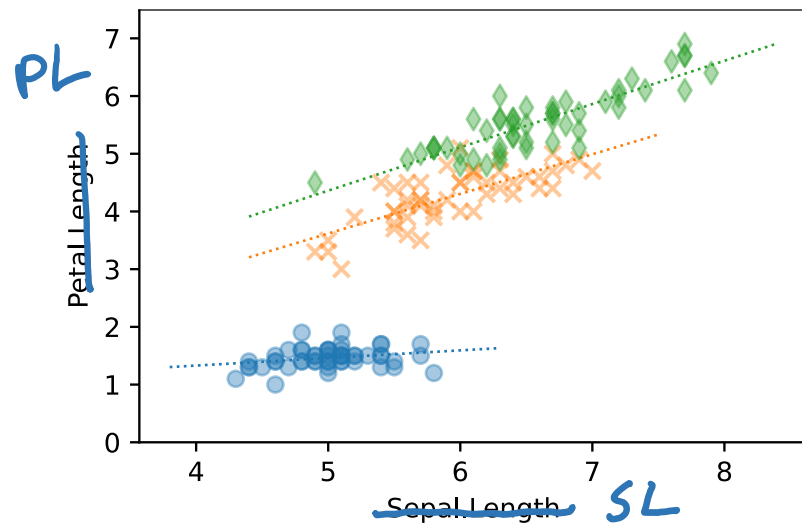
```
class PoissonModel():
    def __init__(self):
        self.l_ = np.nan
    def fit(self, x):
        # Input: x is a numpy vector of integers
        # TODO: set self.l_
```

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?



$$PL \approx \alpha_{\text{spec}} + \beta_{\text{spec}} SL$$

This has 6 unknown pairs:

$$\begin{matrix} \alpha_{\text{seto}} & \alpha_{\text{vers}} & \alpha_{\text{virg}} \\ \beta_{\text{seto}} & \beta_{\text{vers}} & \beta_{\text{virg}} \end{matrix}$$

So my linear model has 6 feature vectors

Row 1: $PL_1 \approx \alpha_{\text{seto}} + \beta_{\text{seto}} SL_1$

Row 2: $PL_2 \approx \alpha_{\text{virg}} + \beta_{\text{virg}} SL_2$
etc.

"one-hot coding"

$$1_{\text{spec}=\text{seto}}$$

$$\begin{matrix} \text{seto} \\ \text{virg} \\ \text{virg} \\ \text{seto} \\ \text{vers} \end{matrix} \begin{bmatrix} PL_1 \\ PL_2 \\ PL_3 \\ PL_4 \\ PL_5 \\ \vdots \end{bmatrix}$$

$$\approx \alpha_{\text{seto}}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$+ \alpha_{\text{virg}}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$+ \alpha_{\text{vers}}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$+ \beta_{\text{seto}}$$

$$\begin{bmatrix} SL_1 \\ 0 \end{bmatrix}$$

$$+ \beta_{\text{virg}}$$

$$\begin{bmatrix} 0 \\ SL_2 \end{bmatrix}$$

$$+ \beta_{\text{vers}}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$1_{\text{spec}=\text{seto}} * \vec{SL}$$

$$PL \approx \alpha_{\text{seto}} 1_{\text{spec}=\text{seto}} + \dots + \beta_{\text{seto}} (1_{\text{spec}=\text{seto}} * \vec{SL}) + \dots$$

I'm using numpy's notation for vector algebra:

- $\vec{1}$ is the constant vector $[1,1,1,1,1]$
- $\overrightarrow{\text{spec}}$ is a vector from the dataset, $["\text{seto}", "\text{virg}", "\text{virg}", "\text{seto}", \dots]$
- $f(\vec{x})$ means "apply the function to each element of $\vec{x}$ "
- $1_{\overrightarrow{\text{spec}}=\text{"seto"}}$ means "apply the indicator to each element of $\overrightarrow{\text{spec}}$ "
- $\overrightarrow{\text{SL}} * 1_{\overrightarrow{\text{spec}}=\text{"seto"}}$ uses element-wise multiplication

```
import sklearn.linear_model
```

```
species, SL, PL = iris['Species'], iris['Petal.Length'], iris['Sepal.Length']
```

```
species_levels = ['setosa', 'virginica', 'versicolor']
```

```
i1,i2,i3 = np.where(species==k, 1, 0) for k in species_levels]
```

```
X = np.column_stack([i1, i2, i3, i1*SL, i2*SL, i3*SL])
```

```
m = sklearn.linear_model.LinearRegression(fit_intercept=False)
```

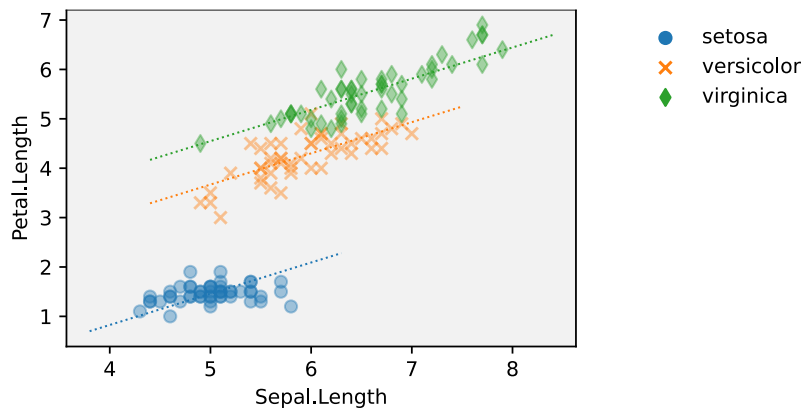
```
m.fit(X, PL)
```

```
m.coef_
```

np.where() is numpy's notation for indicator functions

EXERCISE

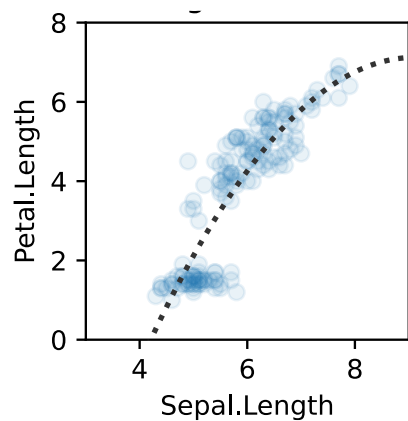
Fit the model with three parallel straight lines.



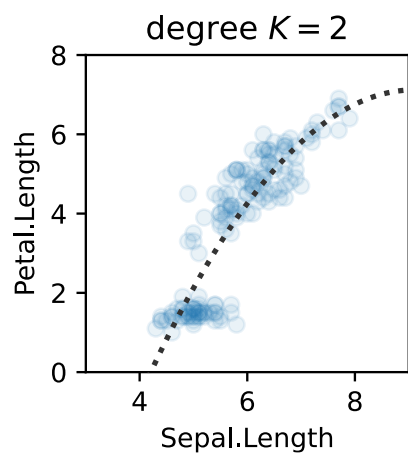
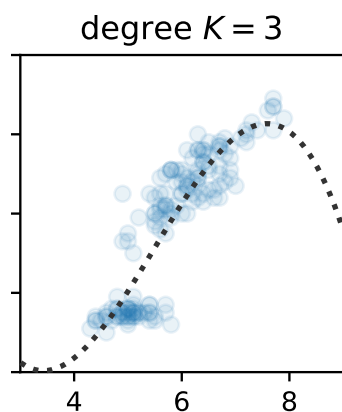
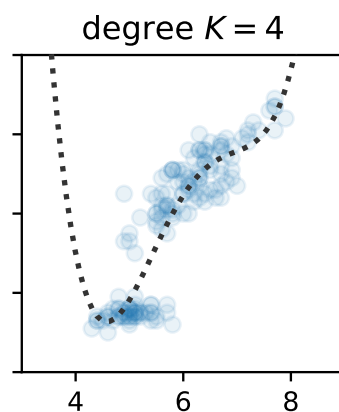
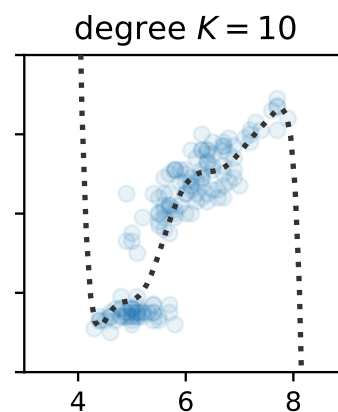
$$PL \approx \alpha_{spec} + \beta SL$$

$$\begin{aligned} \vec{PL} = & \alpha_{seto} \vec{1}_{spec=seto} \\ & + \alpha_{vers} \vec{1}_{\dots} \\ & + \alpha_{virg} \vec{1}_{\dots} \\ & + \beta \vec{SL} \end{aligned}$$

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?

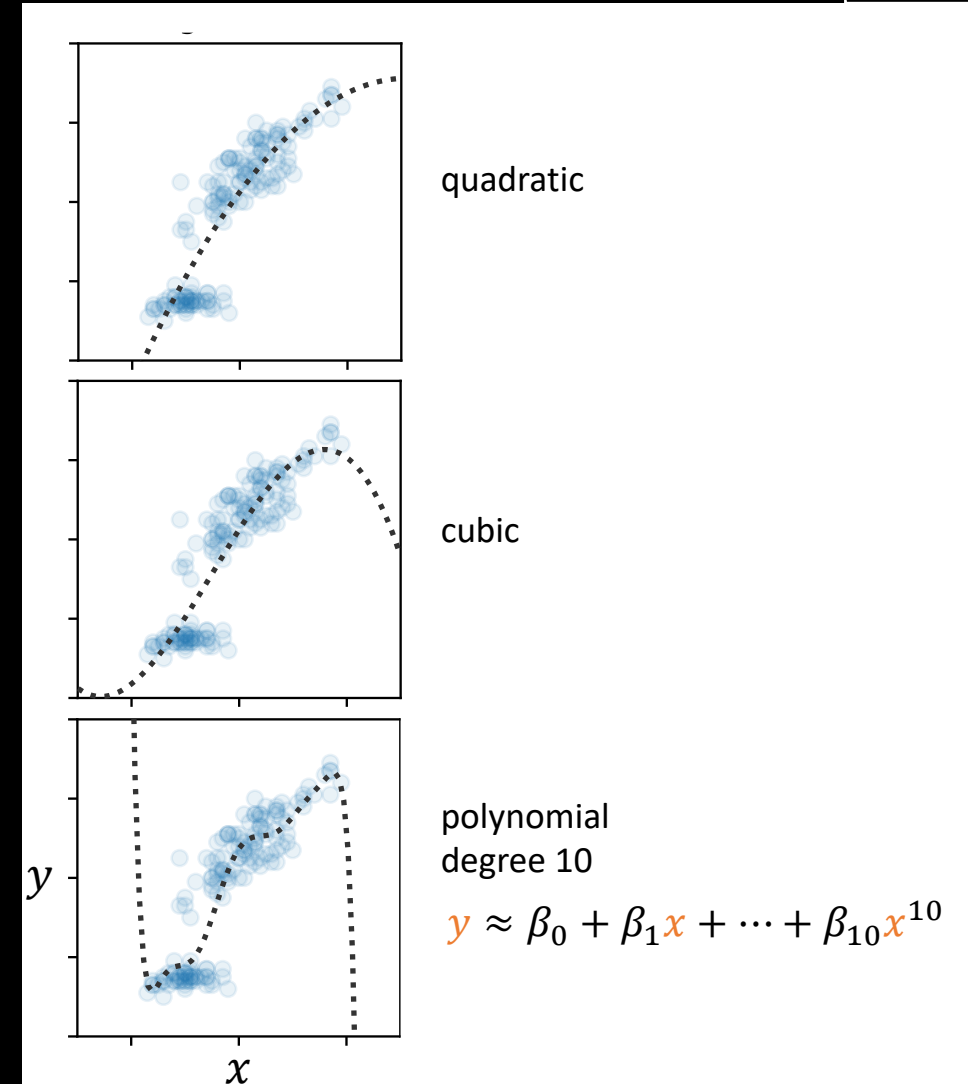
(seeing as the more features we add, the better we can fit the dataset)

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

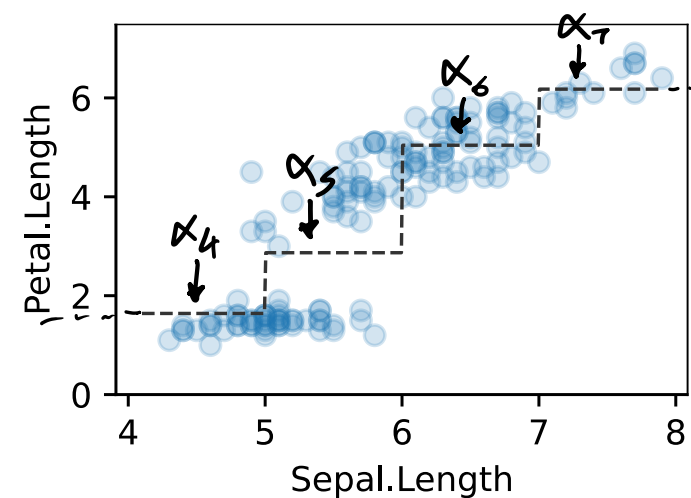
Only add in features that you (as a scientist) believe are relevant.

Or, evaluate your model on a holdout set.

[IA MLRD, Data Science §4]



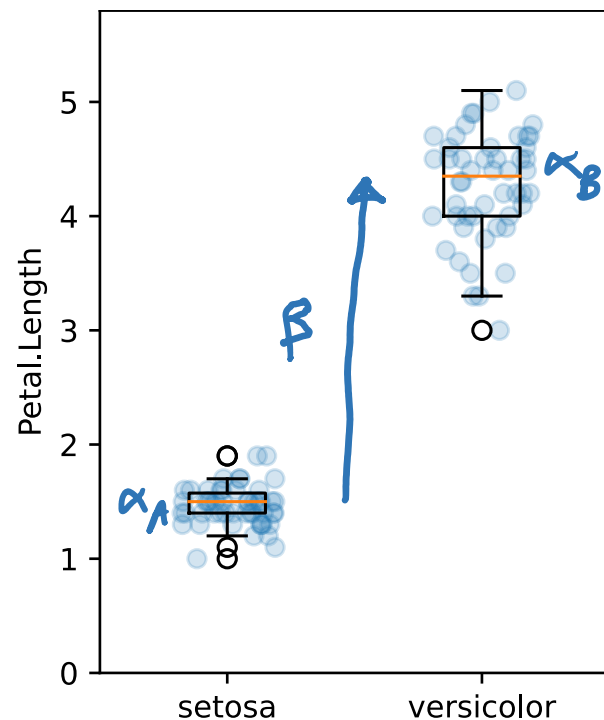
NON-LINEAR RESPONSE via one-hot coding



$$PL \approx \alpha_4 1_{\underbrace{[SL] \leq 4}} + \alpha_5 1_{\underbrace{[SL] = 5}} + \alpha_6 1_{\underbrace{[SL] = 6}} + \alpha_7 1_{\underbrace{[SL] \geq 7}}$$

e.g. for an obs. with $SL = 5.3$ my model predicts $PL \approx \alpha_5$
 $SL = 2.7$ $PL \approx \alpha_4$

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?

Version 1:

$$x \approx \begin{cases} \alpha_A & \text{if } j=A \\ \alpha_B & \text{if } j=B \end{cases}$$

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

condition j	response x
$\left. \begin{matrix} A \\ A \\ A \\ \vdots \\ A \end{matrix} \right\} m$	$\begin{matrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{matrix}$
$\left. \begin{matrix} B \\ \vdots \\ B \end{matrix} \right\} n$	$\begin{matrix} b_1 \\ \vdots \\ b_n \end{matrix}$

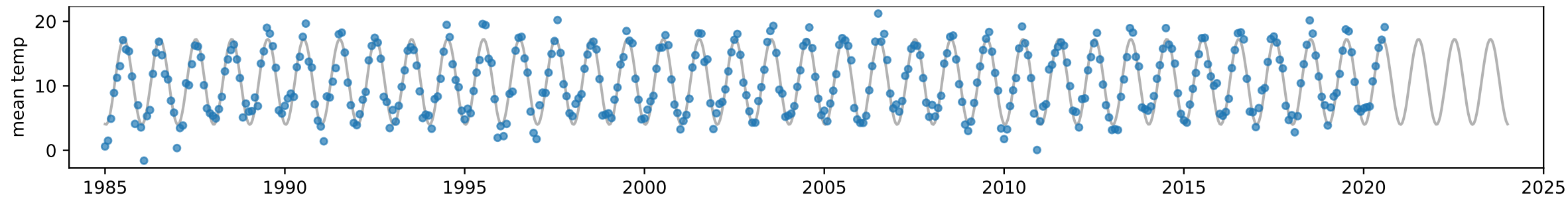
Version 2:

$$\vec{x} \approx \alpha + \beta \mathbf{1}_{\vec{j}=B}$$

$$x \approx \begin{cases} \alpha & \text{if } j=A \\ \alpha + \beta & \text{if } j=B \end{cases}$$

In this version, the quantity I'm interested in (the group difference β) appears directly as a parameter in the model, so my fit reports it directly.

PERIODIC PATTERN



We'd like to fit the model:

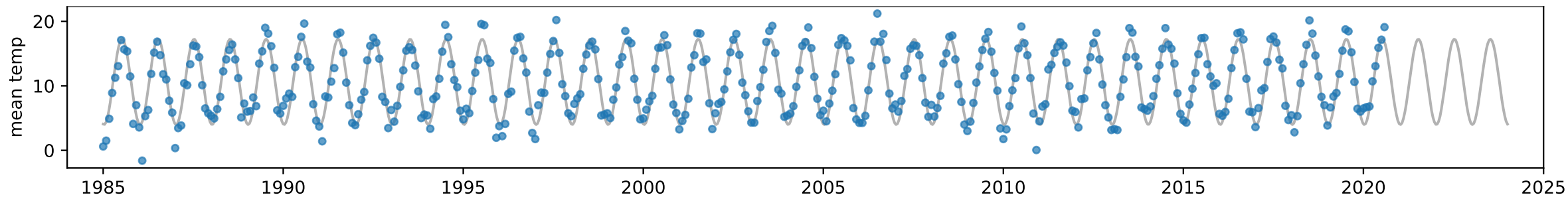
$$\text{temp} \approx \alpha + \beta \sin(2\pi t + \varphi)$$

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

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

Instead, we could just brute-force optimize it with `scipy.optimize.fmin`.

PERIODIC PATTERN



We'd like to fit the model:

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

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

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

$$= \alpha + \beta_1 \sin(2\pi t) + \beta_2 \cos(2\pi t)$$

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

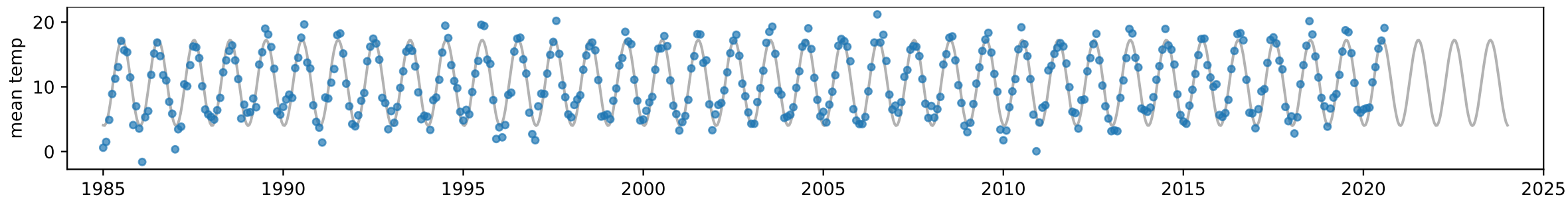
From secondary school trigonometry,

$$\begin{aligned} \sin(A + B) \\ = \sin(A) \cos(B) + \cos(A) \sin(B) \end{aligned}$$

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

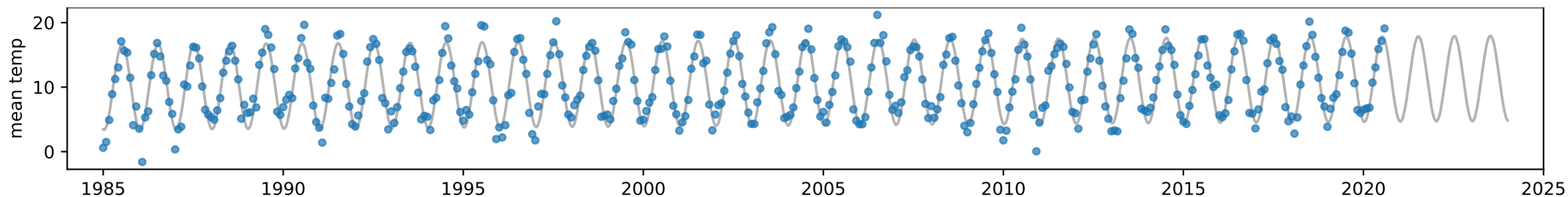
PERIODIC PATTERN

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



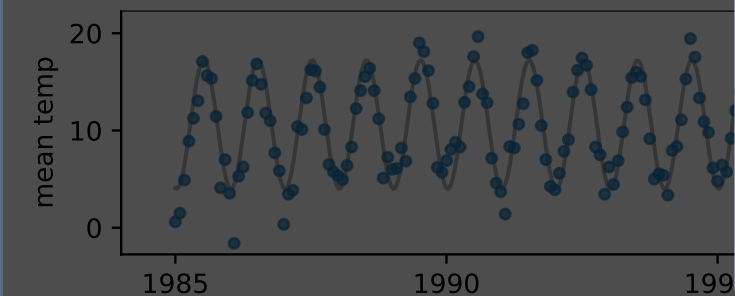
PERIODIC PATTERN + SECULAR TREND

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



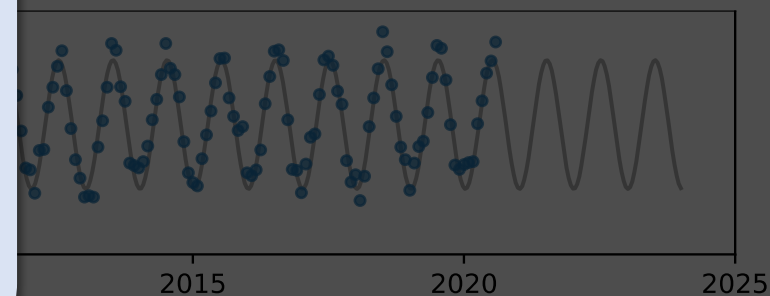
PERIODIC PATTERN

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



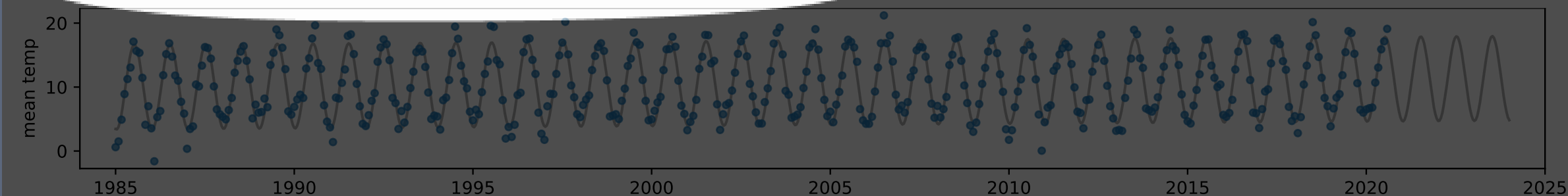
Linear models are easily composable.

Here I added on the “linear trend” feature. We could easily have stuck in the step-function response instead.



PERIODIC PATTERN + SECULAR TREND

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



climate

x +

cl.cam.ac.uk/teaching/2223/DataSci/datasci/ex/climate.html

In [1]:

```
import numpy as np
import pandas
import matplotlib.pyplot as plt
import sklearn.linear_model
π = np.pi
```

Climate dataset challenge

- What is the rate of temperature increase in Cambridge?
- Are temperatures increasing at a constant rate, or has the increase accelerated?
- How do results compare across the whole of the UK?

Your task is to answer these questions using appropriate linear models, and to produce elegant plots to communicate your findings. Please submit a Jupyter notebook, or a pdf. Include explanations of what your models are, and of what your plots show.

The dataset is from <https://www.metoffice.gov.uk/pub/data/weather/uk/climate/>. Code for retrieving the dataset is given at the bottom.

Upload your answers to Moodle by Sunday for presentation / discussion next week