Search

Fork Sign in

Welcome. This is live code! Click the left margin to view or edit.

Kris Sankaran

2

Published Sep 10, 2019

Function Fitting

IFT6758, Fall 2019

Reading: ISLR sections 2.1, 3.2.1, 3.5

?

An Abstraction: Input → Output

Across applications we often want to model an output variable as a function of many inputs,

- Genetic profile → Chance of developing disease
- Person's characteristics → Whether they'll vote

- Person's characteristics → Whether they'll vote
- Marketing plan → Total sales amount
- Image pixel values → What's in the image

An Abstraction: Input → Output

Across applications we often want to model an output variable as a function of many inputs,

- Genetic profile → Chance of developing disease
- Person's characteristics → Whether they'll vote
- Marketing plan → Total sales amount
- Image pixel values → What's in the image

$$x_i = (x_{i1}, \ldots, x_{ip}) \leftarrow \text{all the inputs}$$

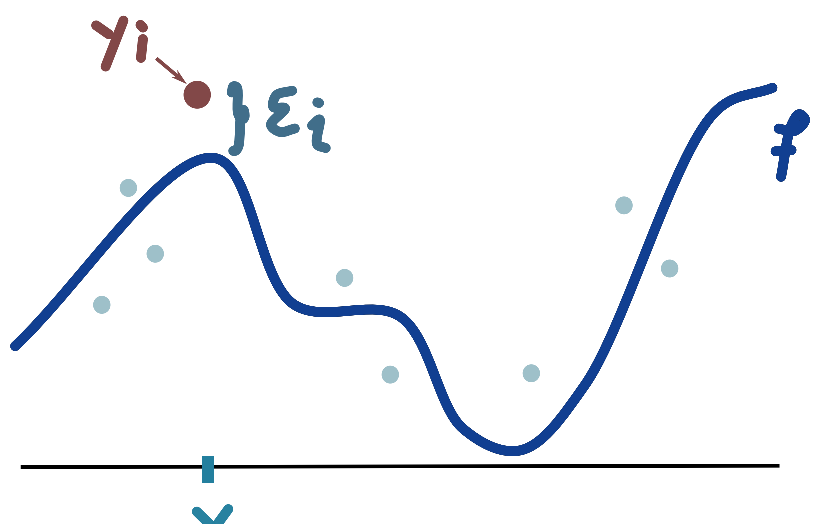
$$y_i = f(x_i) \leftarrow \text{inputs to output relationship}$$

$y_i = f(x_i) \leftarrow$ inputs to output relationship

A Picture

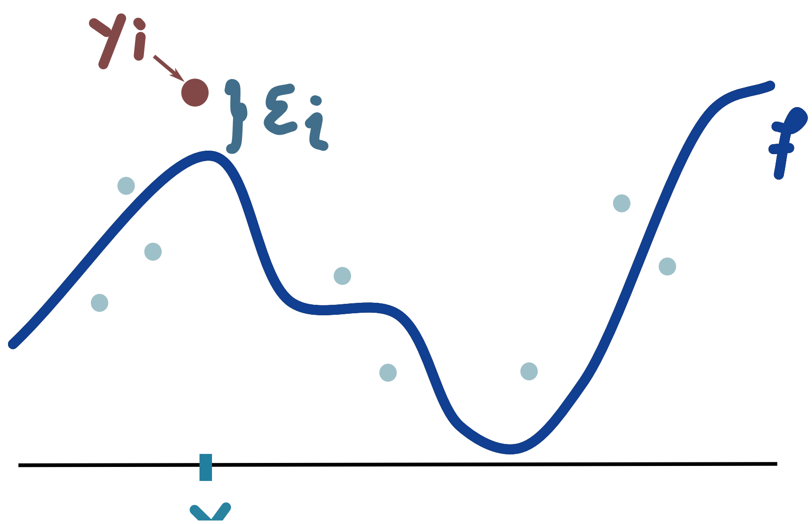
We allow for y_i to be a different from f, perturbed by some observation-specific noise ϵ_i,

$$y_i = f(x_i) + \epsilon_i$$



Why this decomposition?

- f describes systematic variation in y_i, as a function of observed inputs.
- ϵ_i reflects variation whose source is unknown to us. Consider coin tosses.



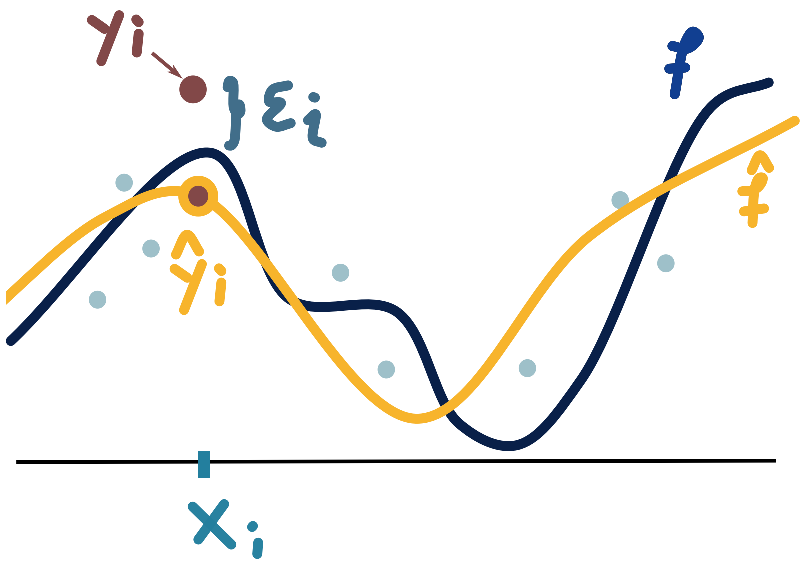
x_i

Why are we doing this?

- Why not just visualize the data we have?
- Reason 1: **Prediction**
  - We may want the y_i corresponding to an input x_i
  - Inputs may be much easier to collect than outputs
- Reason 2: **Inference**
  - We may care about the form of f, for personal understanding
  - e.g., is a particular input relevant at all?
- We want quantitative estimates, not visual summaries

Estimates and Predictions

- In reality, we won't know f
- We estimate it from data and call the result $\hat{f}$
- To predict y_i for some input x_i, we'd use $\hat{y}_i = \hat{f}(x_i)$



Sources of Error

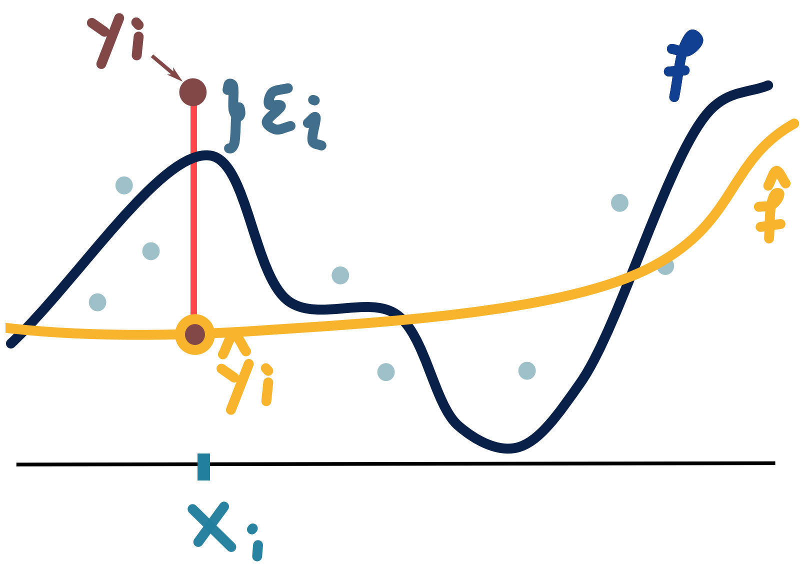
This process introduces two sources of error,

- Approximation error: $\hat{f}$ isn't close to f
  - This error is *reducible* (use a better algorithm)
- Irreducible error: y_i isn't close to $f(x_i)$
  - Incur this error even if f were perfectly known

Sources of Error

This process introduces two sources of error,

- **Approximation error: $\hat{f}$ isn't close to f**
  - This error is *reducible* (use a better algorithm)
- Irreducible error: y_i isn't close to $f\left(x_i\right)$
  - Incur this error even if f were perfectly known

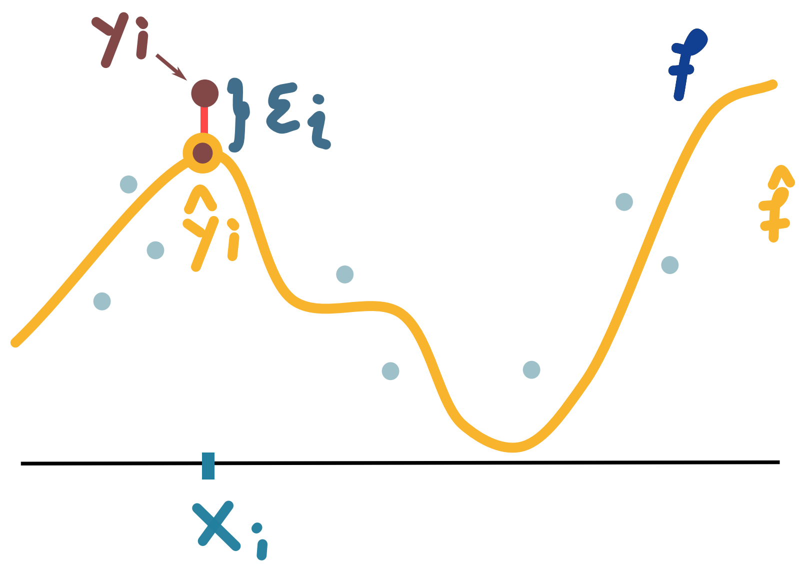


?

Sources of Error

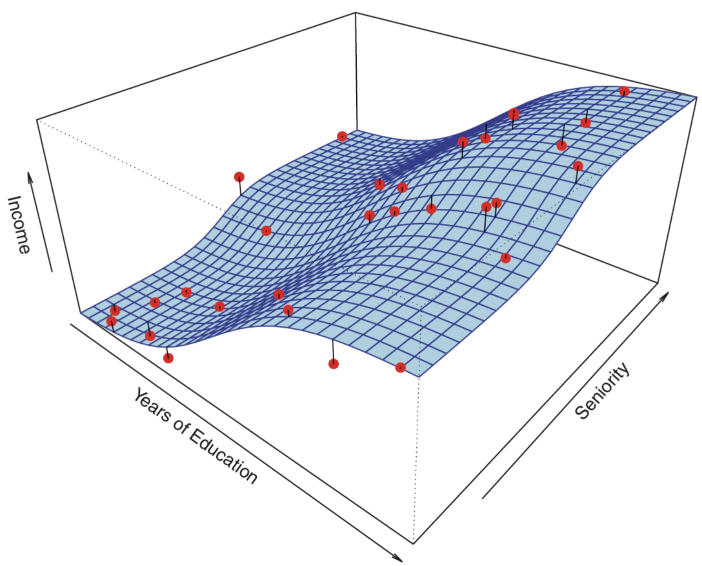
This process introduces two sources of error,

- Approximation error: $\hat{f}$ isn't close to f
  - This error is *reducible* (use a better algorithm)
- **Irreducible error: y_i isn't close to $f\left(x_i\right)$**
  - Incur this error even if f were perfectly known



Extending the Picture

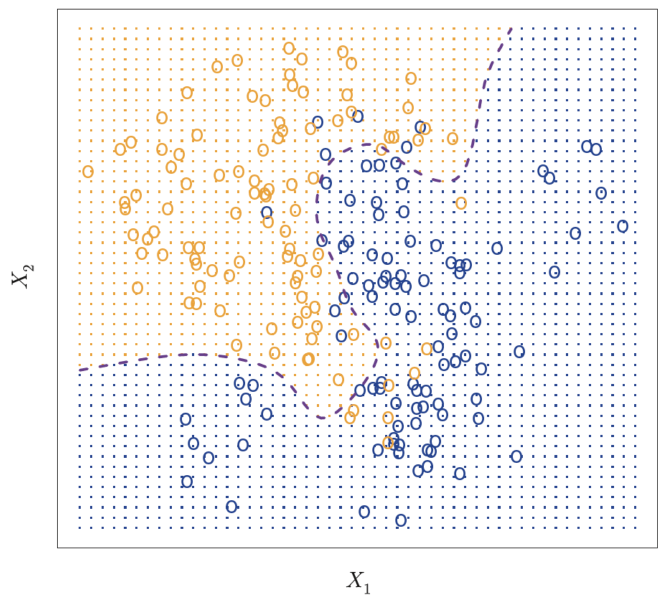
- The abstraction is much more general
- It applies to high-dimensional x_i and general y_i



Here, x_i are two dimensional.

Extending the Picture

- The abstraction is much more general
- It applies to high-dimensional x_i and general y_i



Here, the response y_i is binary (blue or orange).

How to find $\hat{f}$?

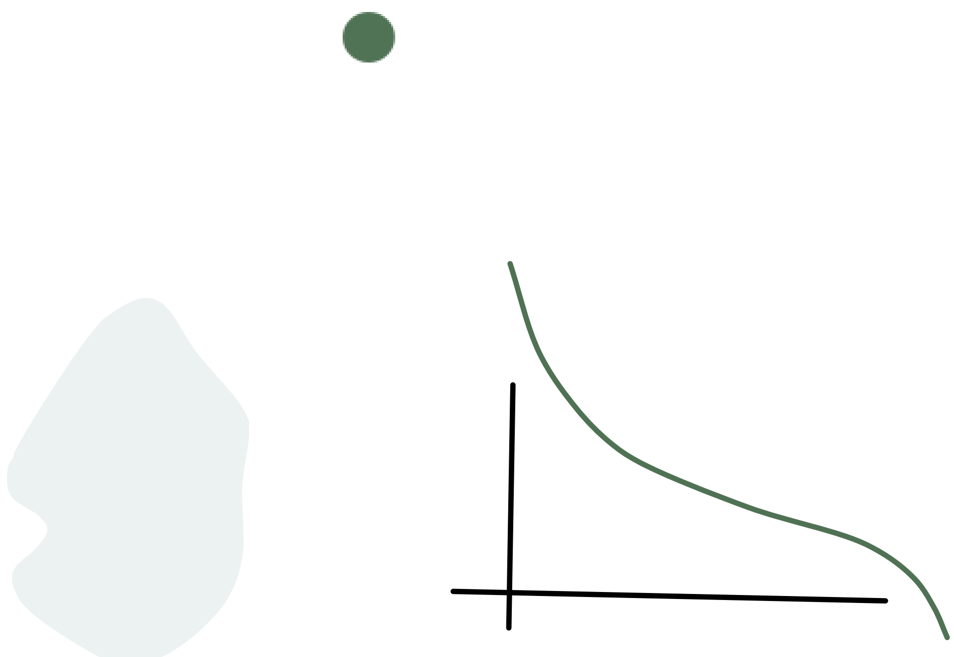
Generally have two steps,

- Propose a model family $\mathcal{F}$
  - E.g., set of all linear functions of x_i
- Define a procedure to choose $\hat{f} \in \mathcal{F}$ based on the data
  - E.g., the choice $\hat{f}$ that minimizes $\sum_i \left(y_i - \hat{f}\left(x_i\right)\right)^2$

How to find $\hat{f}$?

Generally have two steps,

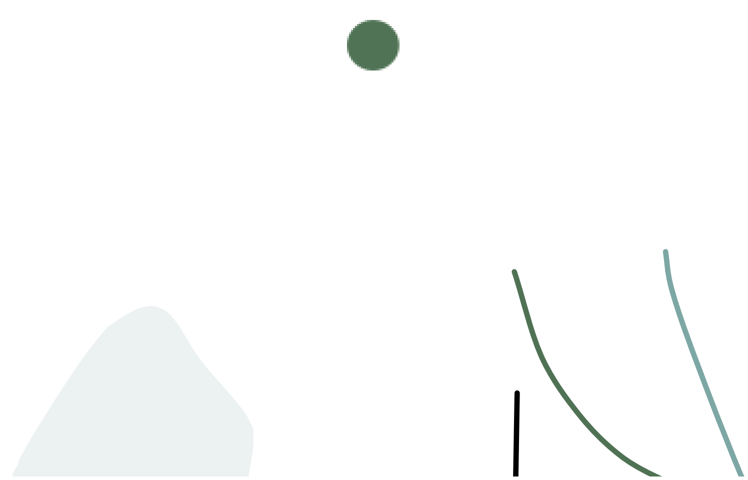
- Propose a model family $\mathcal{F}$
  - E.g., set of all linear functions of x_i
- Define a procedure to choose $\hat{f} \in \mathcal{F}$ based on the data
  - E.g., the choice $\hat{f}$ that minimizes $\sum_i \left(y_i - \hat{f}\left(x_i\right)\right)^2$

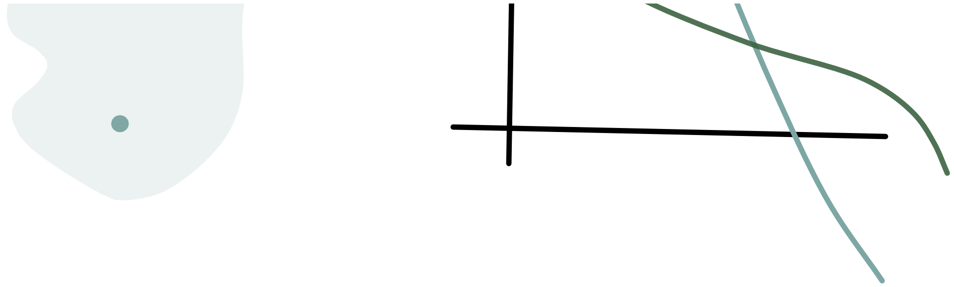


How to find $\hat{f}$?

Generally have two steps,

- Propose a model family $\mathcal{F}$
    - E.g., set of all linear functions of x_i
- Define a procedure to choose $\hat{f} \in \mathcal{F}$ based on the data
    - E.g., the choice $\hat{f}$ that minimizes $\sum_i \left(y_i - \hat{f}\left(x_i\right)\right)^2$

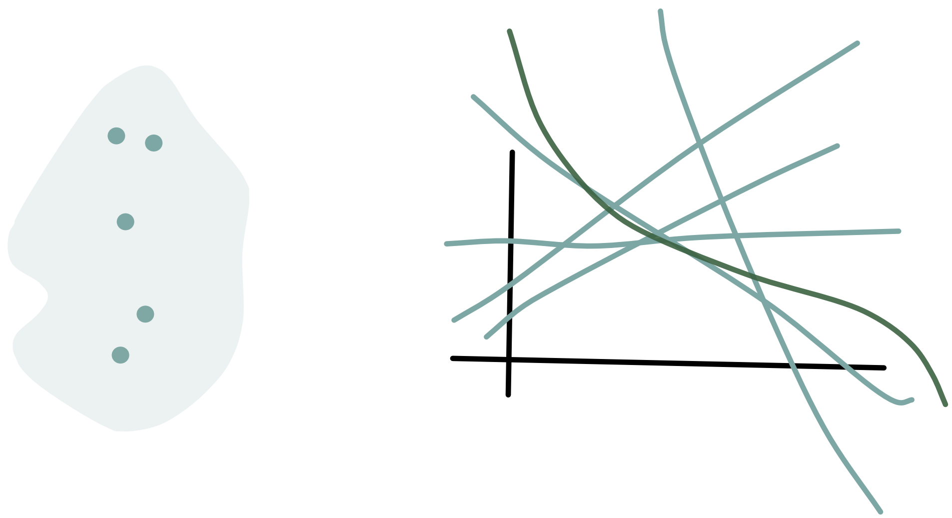




How to find $\hat{f}$?

Generally have two steps,

- Propose a model family $\mathcal{F}$
  - E.g., set of all linear functions of x_i
- Define a procedure to choose $\hat{f} \in \mathcal{F}$ based on the data
  - E.g., the choice $\hat{f}$ that minimizes $\sum_i \left(y_i - \hat{f}\left(x_i\right)\right)^2$



?

Why ever use smaller $\mathcal{F}$'s?

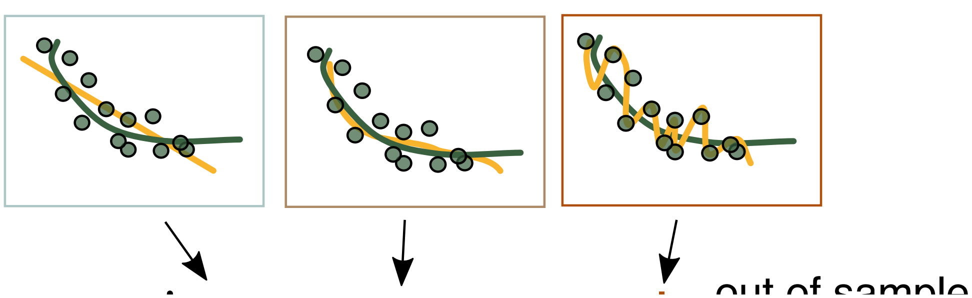
- There is a bias-variance tradeoff
- Finding the best function in a large class $\mathcal{F}$ can be hard
  - High variance: Different samples x_i, y_i might result in

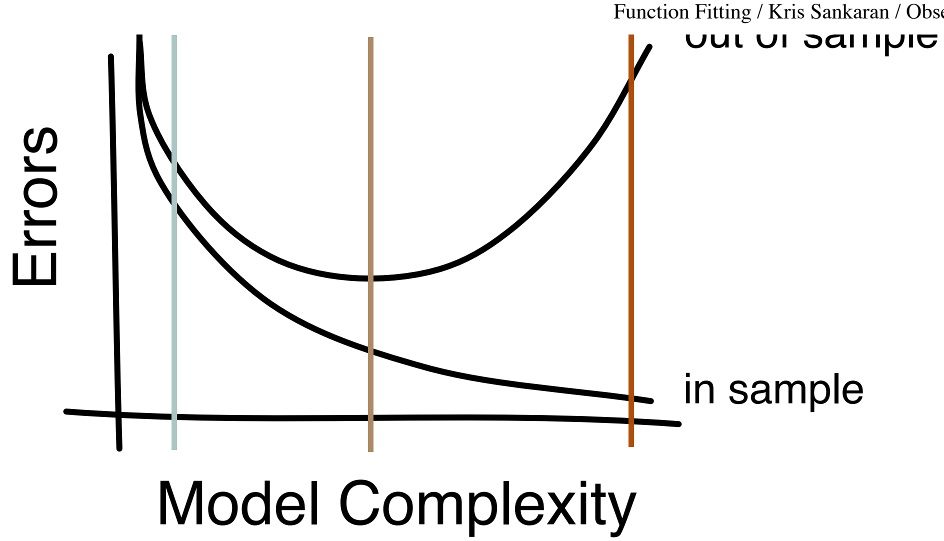
very different $\hat{f}$, even when f hasn't changed

- This variance gets worse the higher-dimensional your x_i are (the "curse of dimensionality")
- Incurring some bias, for the sake of better stability, can lead to overall better predictions

Why ever use smaller $\mathcal{F}$'s?

- Ultimately, you want your model to perform well on out-of-sample data



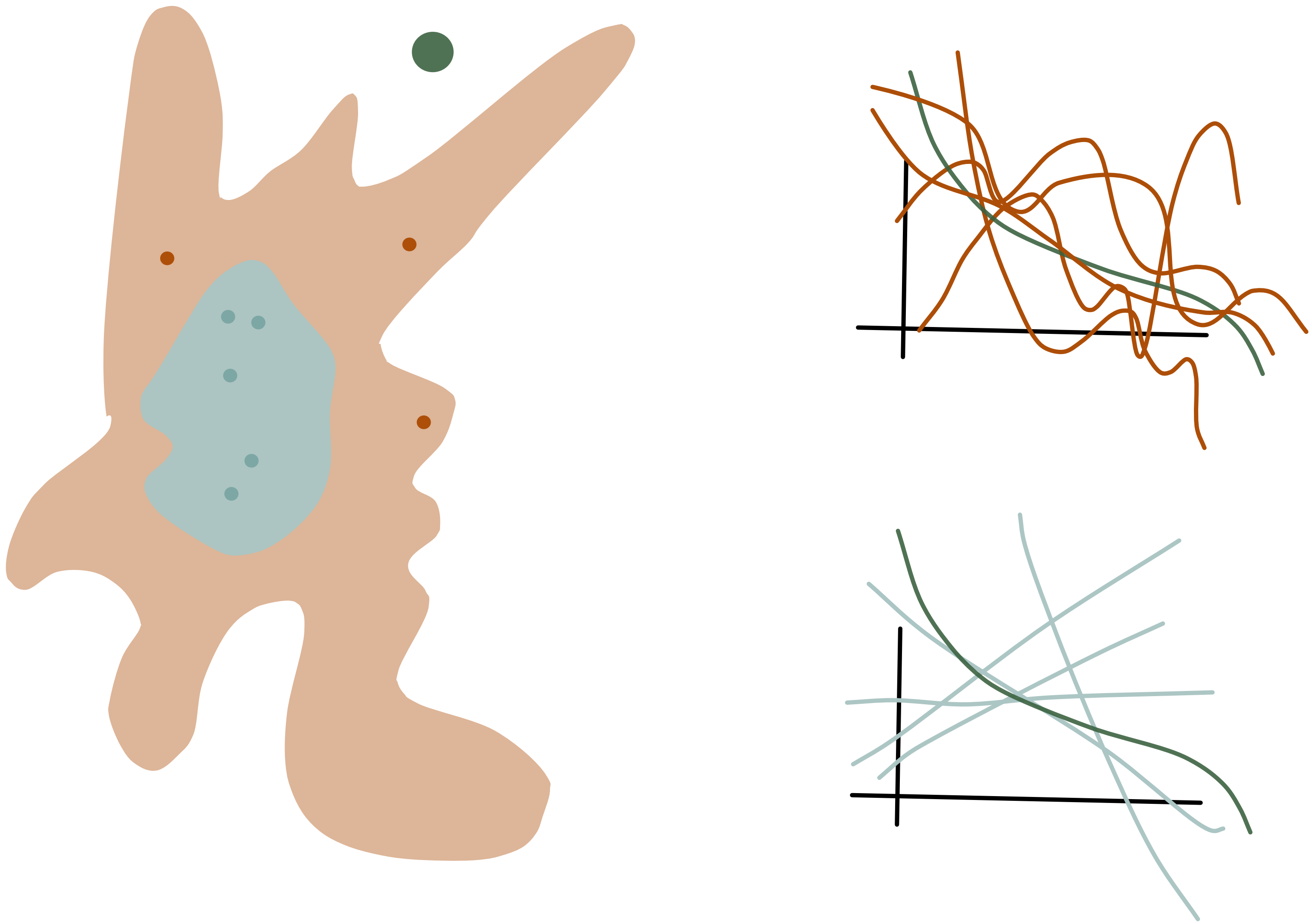


?

Why ever use smaller $\mathcal{F}$'s?

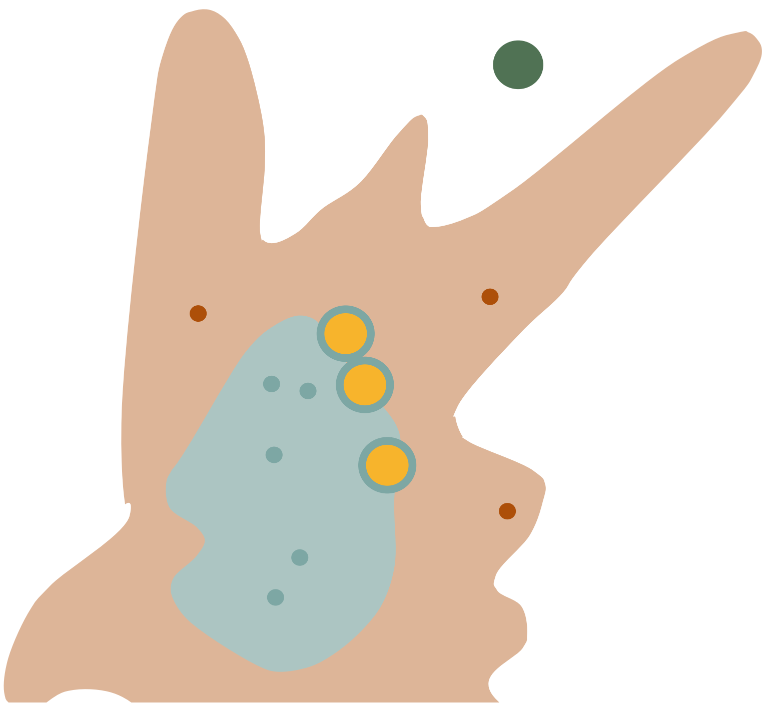
- There is a bias-variance tradeoff
- Finding the best function in a large class $\mathcal{F}$ can be hard
  - High variance: Different samples x_i, y_i might result in very different $\hat{f}$, even when f hasn't changed
  - Incurring some bias, for the sake of better stability, can

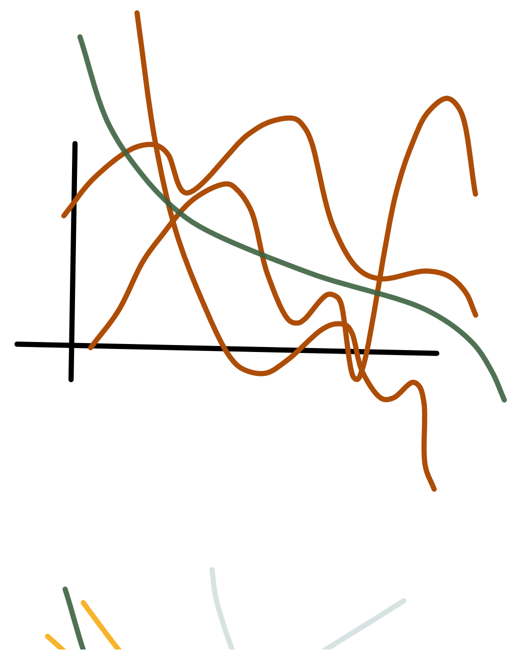
lead to overall better predictions

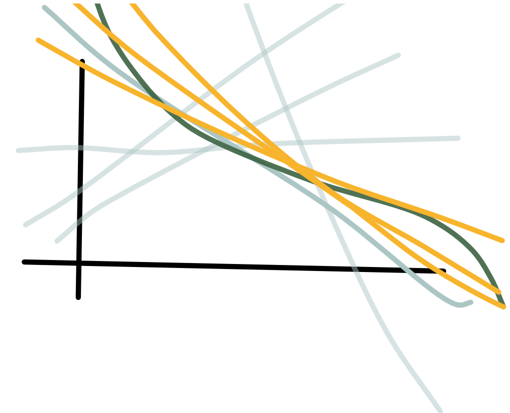


Why ever use smaller $\mathcal{F}$'s?

- There is a bias-variance tradeoff
- Finding the best function in a large class $\mathcal{F}$ can be hard
  - High variance: Different samples x_i, y_i might result in very different $\hat{f}$, even when f hasn't changed
  - Incurring some bias, for the sake of better stability, can lead to overall better predictions

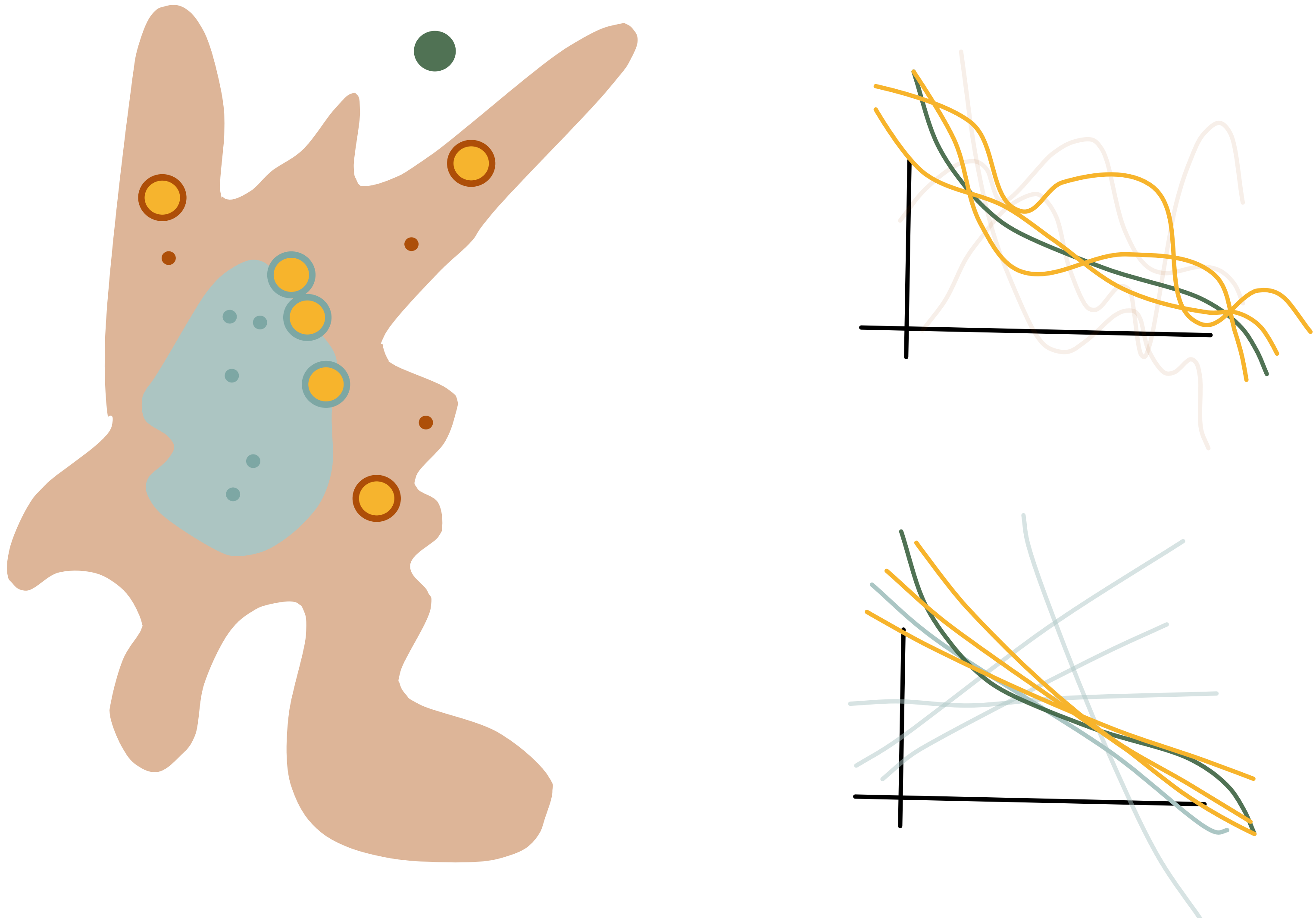






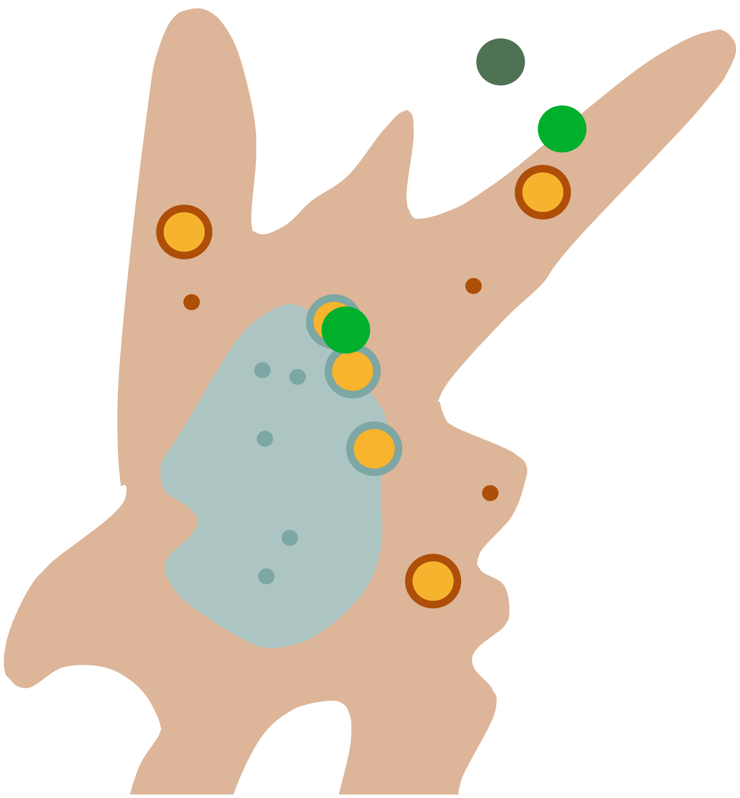
Why ever use smaller $\mathcal{F}$'s?

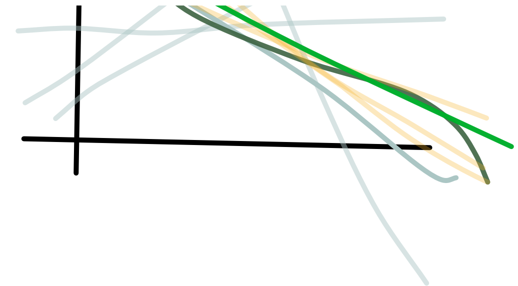
- There is a bias-variance tradeoff
- Finding the best function in a large class $\mathcal{F}$ can be hard
    - High variance: Different samples x_i, y_i might result in very different $\hat{f}$, even when f hasn't changed
    - Incurring some bias, for the sake of better stability, can lead to overall better predictions



Why ever use smaller $\mathcal{F}$'s?

- There is a bias-variance tradeoff
- Finding the best function in a large class $\mathcal{F}$ can be hard
    - High variance: Different samples x_i, y_i might result in very different $\hat{f}$, even when f hasn't changed
    - Incurring some bias, for the sake of better stability, can lead to overall better predictions





Discussion Questions

- What are the advantages of a more vs. a less flexible regression model? When would you prefer one vs. the other?
  - How does your answer depend on the input dimension of the x_i?
  - How does your answer depend on the sample size?

Model Flexibility -- Takeaways

This tradeoff can be summarized with a few takeaways,

- If you don't have too many samples, you should prefer a simpler model
- If you have many samples, you can afford a more complex model
- We'll need some sort of mechanism to tell which regime we're in

Examples of Model Families

- Before discussing specific algorithms in detail, let's get a feel for how different model families look like
- We'll dive into their details in the next few lectures
- We're using the `Advertising` dataset (how does advertising affect sales?)

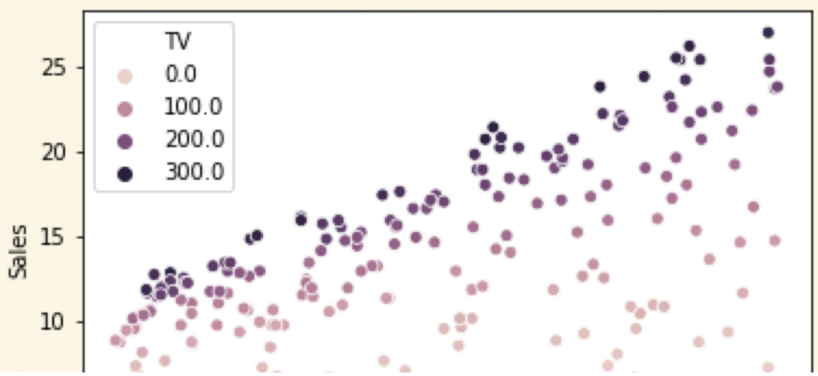
```
In [14]:
ads = pd.read_csv("https://gist.githu
d51e9aa149768/advertising.csv").iloc[
ads
Out [14]:
        TV  Radio  Newspaper  Sales
0    230.1   37.8       69.2   22.1
1     44.5   39.3       45.1   10.4
2     17.2   45.9       69.3    9.3
3    151.5   41.3       58.5   18.5
4    180.8   10.8       58.4   12.9
..     ...    ...        ...    ...
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1    9.7
197  177.0    9.3        6.4   12.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   13.4

[200 rows x 4 columns]
```

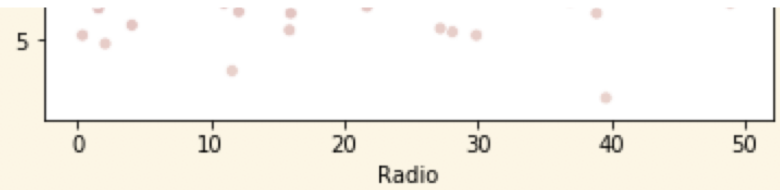
?

Examples of Model Families

- Before discussing specific algorithms in detail, let's get a feel for how different model families look like
- We'll dive into their details in the next few lectures
- We're using the `Advertising` dataset (how does advertising affect sales?)

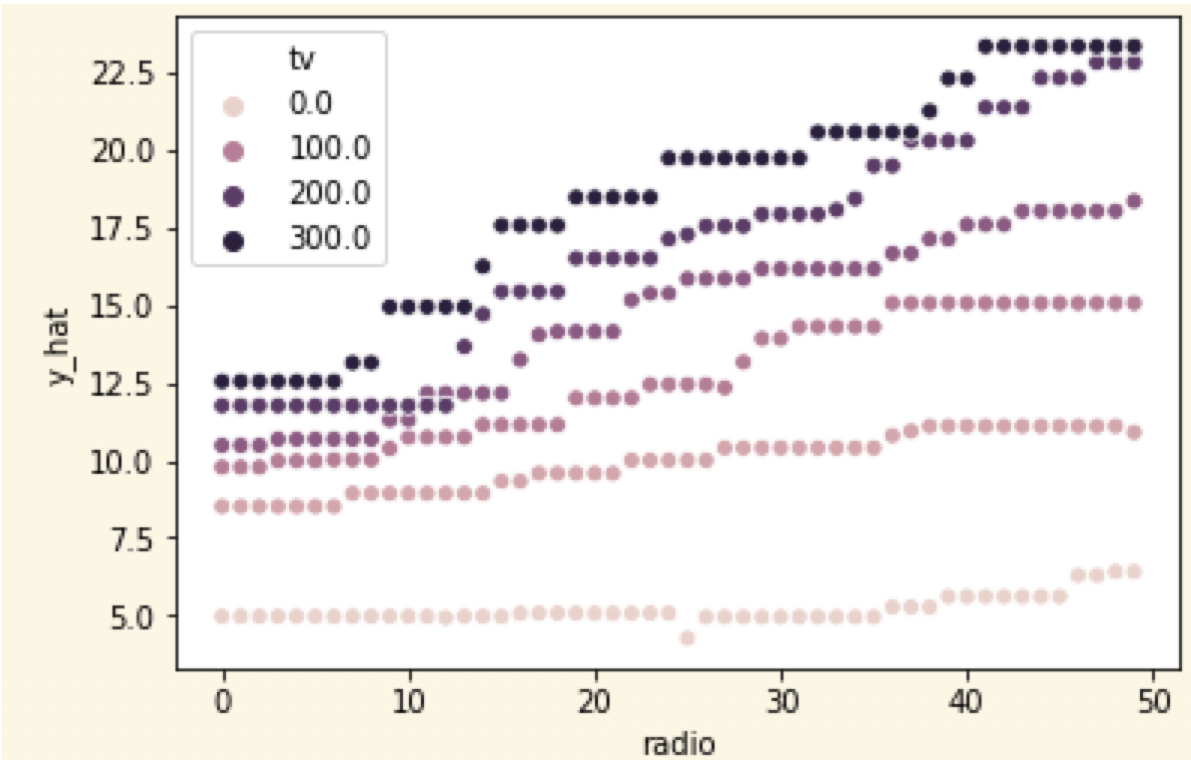
```
In [12]:
sns.scatterplot(x="Radio", y="Sales", hue="TV", data=ads)
Out [12]:
<matplotlib.axes._subplots.AxesSubplot at 0x11610f470>
```



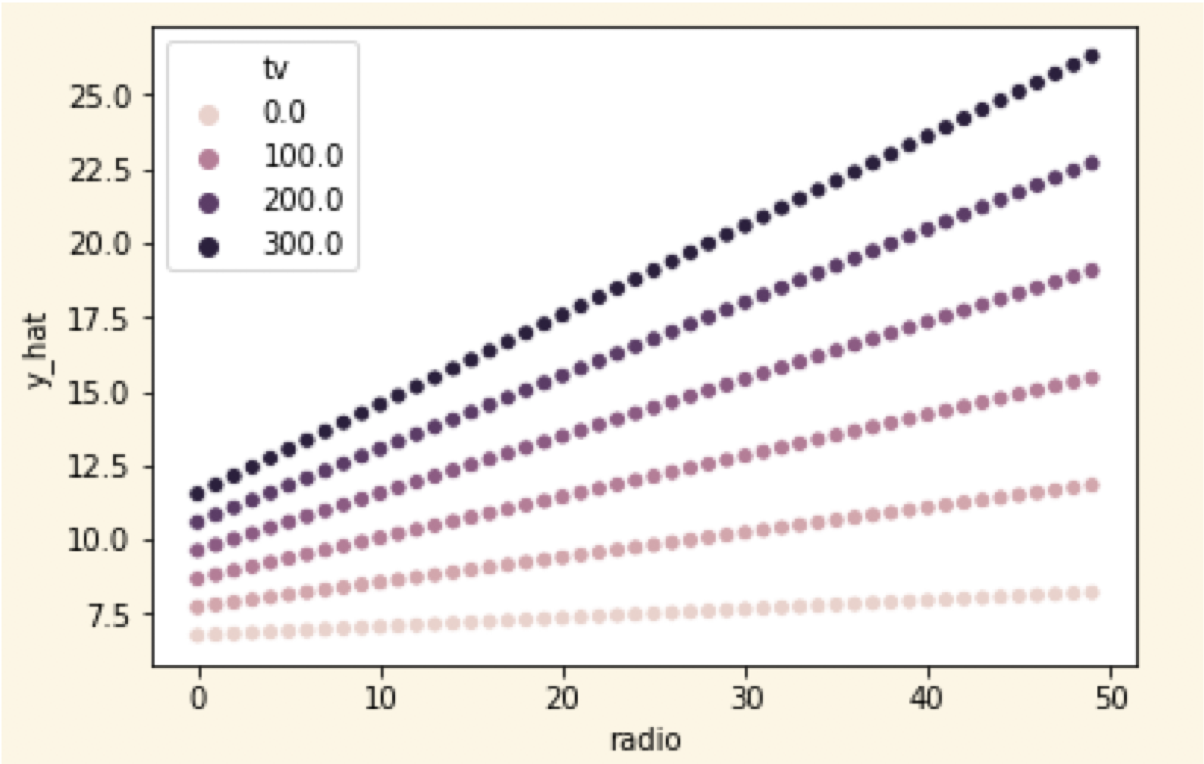


Linear Regression

K-Nearest Neighbors

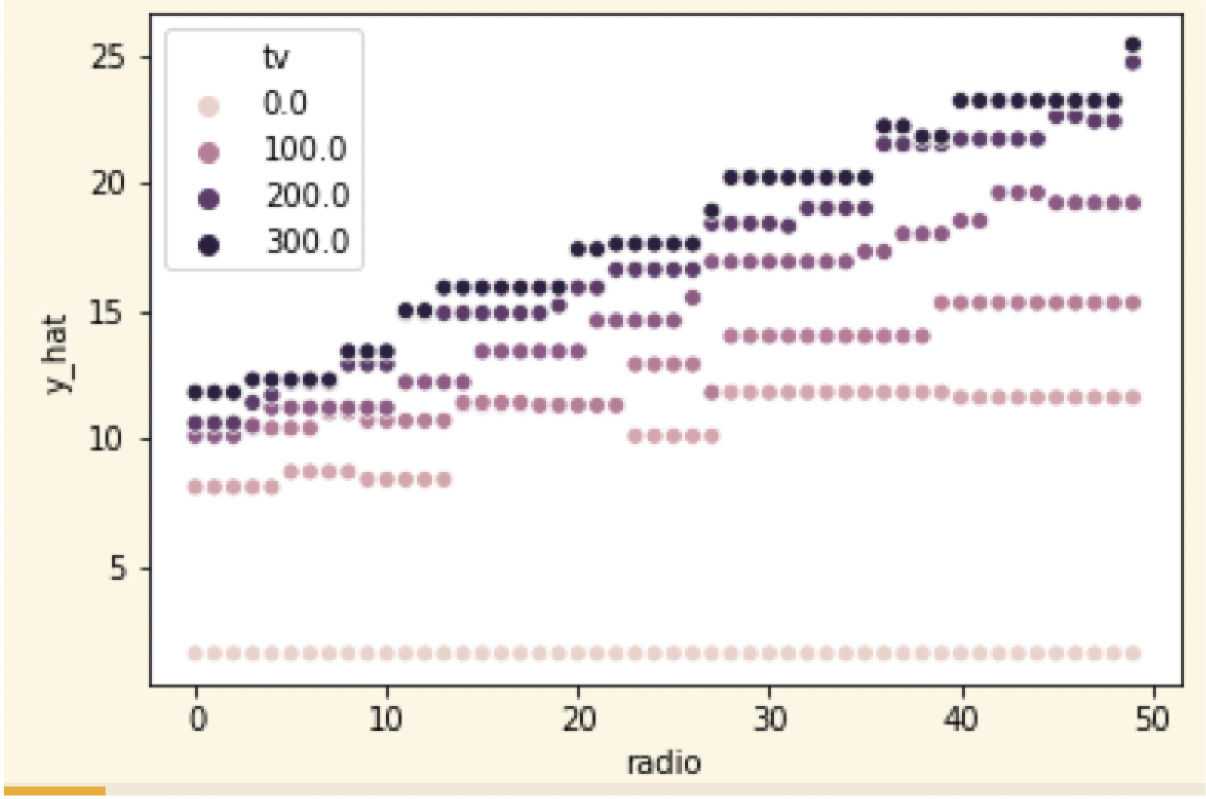


Linear Regression with Interaction



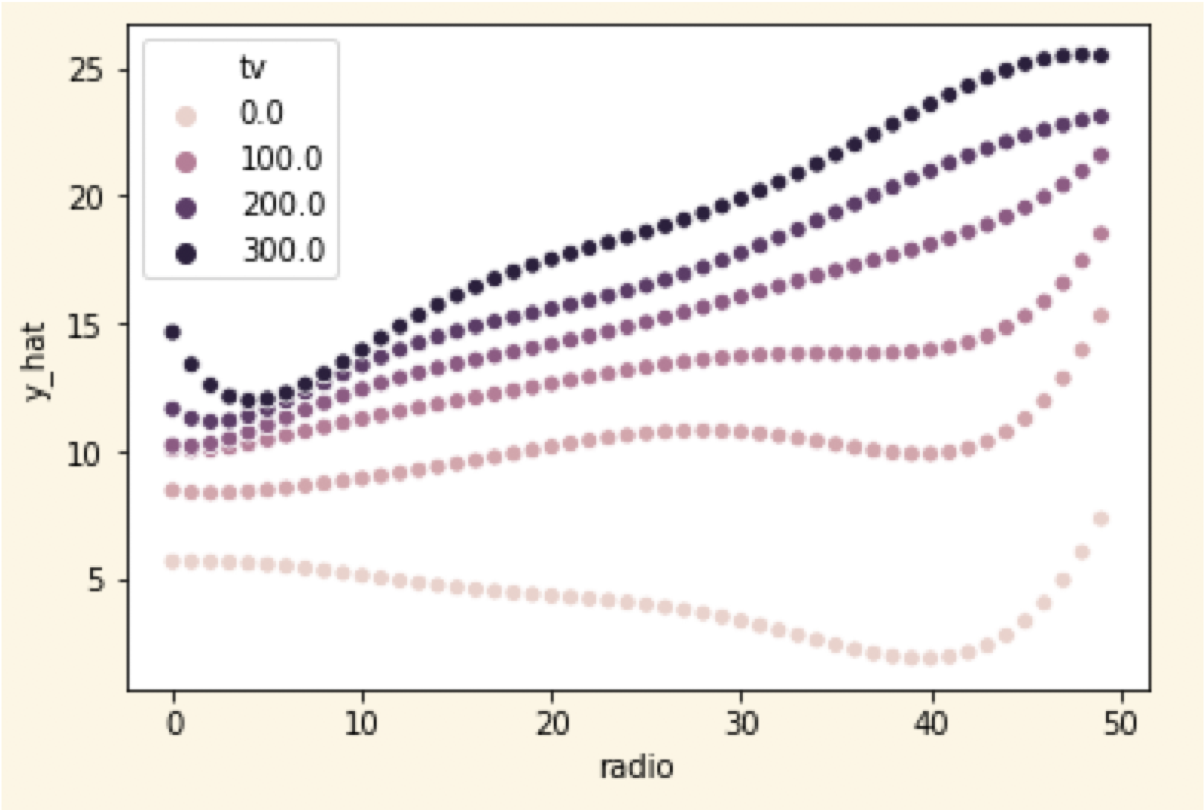
Notice that each line has a different slope.

Trees



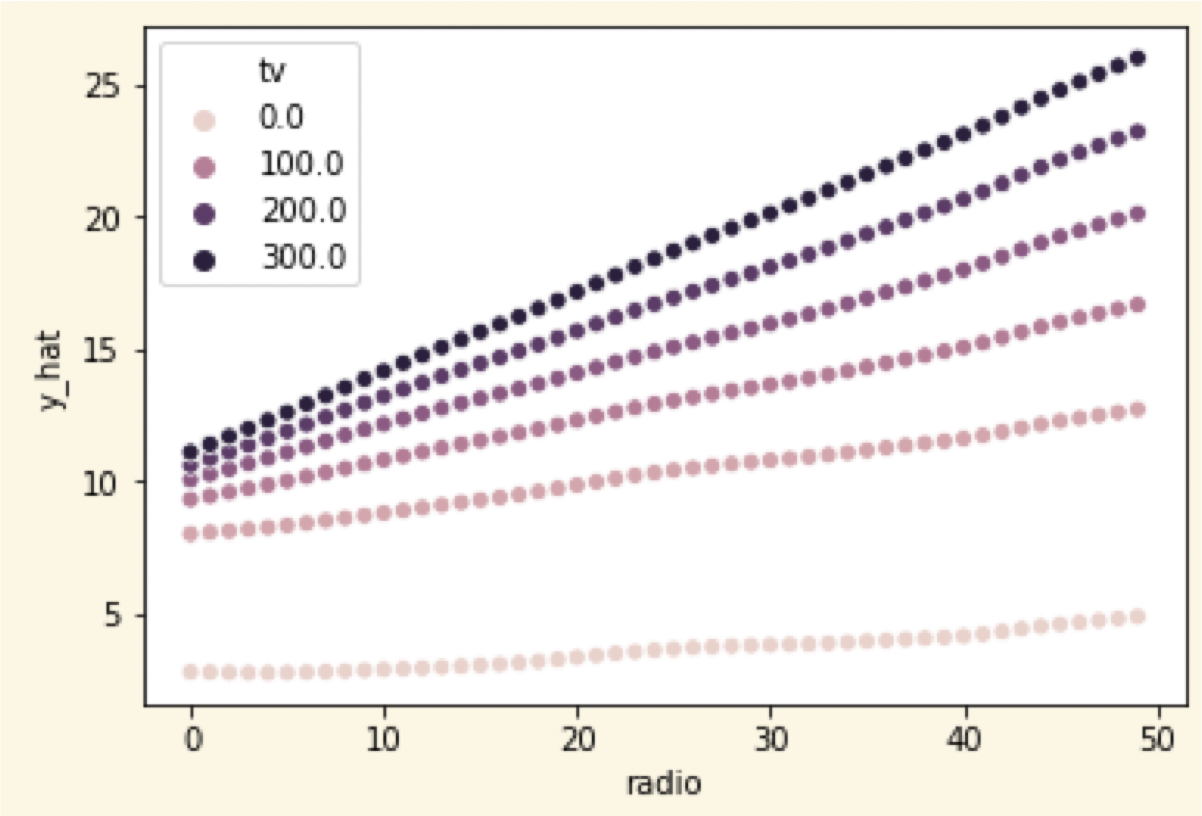
This vaguely resembles the KNN fit.

Linear Regression with Nonlinear Basis



A *linear* combination of *nonlinear* functions will be *nonlinear*.

Generalized Additive Modes



It looks quite similar to the interaction model, but is a little more wiggly.

Discussion

Which of the function families above would you guess is parametric? Which would you guess is nonparametric?

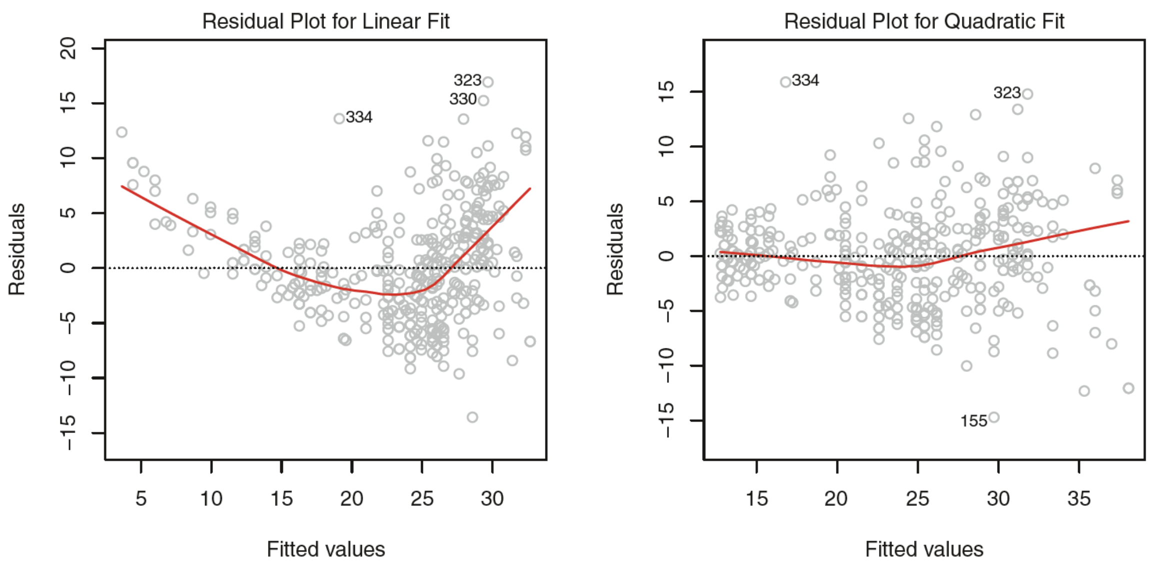
(I don't expect you to know the answer yet, but make an educated guess based on the pictures)

Post-training Analysis

- There's a certain set of checks you should always do after you fit a model, no matter what family it is
  - Yes, even deep learning models
- You can do better than looking at the validation loss
- Residual analysis, error modeling, outliers, high-leverage

Residual Analysis

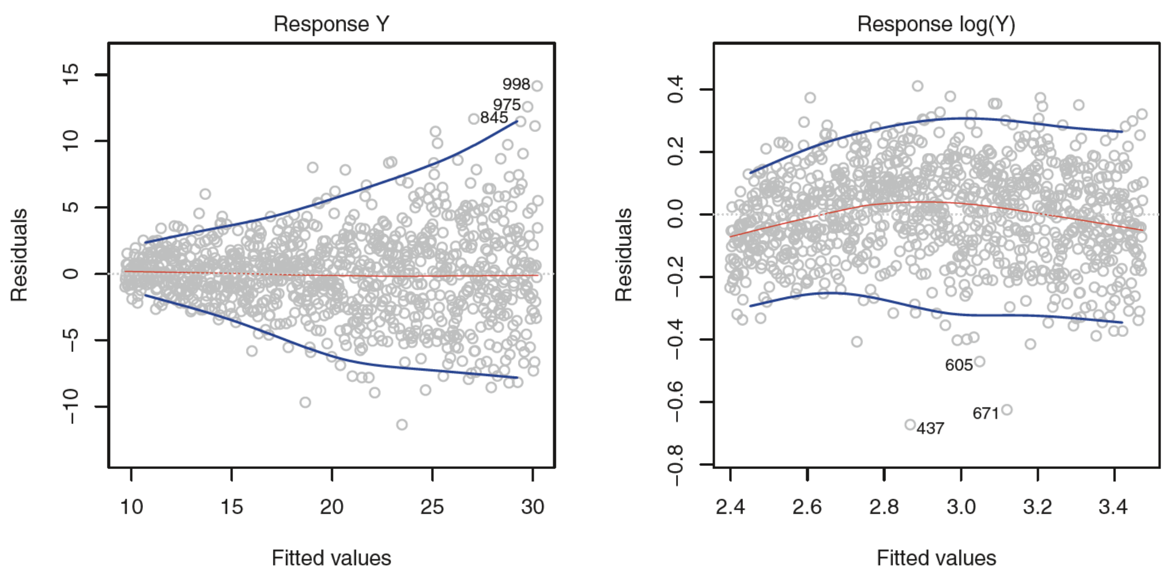
- Make a histogram of errors $e_i = y_i - \hat{y}_i$
- Plot them against a few input variables
- If you notice systematic variation in them, this is information you can squeeze into f



Residual Analysis

Residual Analysis

- Make a histogram of errors $e_i = y_i - \hat{y}_i$
- Plot them against a few input variables
- If you notice systematic variation in them, this is information you can squeeze into f



Error Modeling

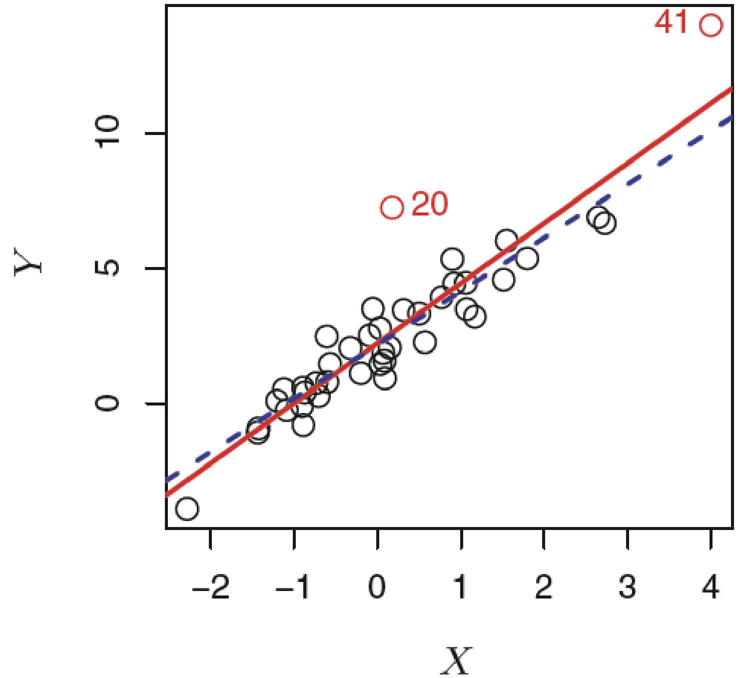
- You can use models to seek out systematic variation in e_i
- Cluster the x_i associated with large errors $|e_i|$
- Train a model with e_i as a response
  - E.g., if you use a tree, the first few split variables still have information that you haven't made use of (we'll describe trees next lecture)

?

Outliers and Leverage

- Look for outliers either in the x or y directions

- High leverage points are those that, if they were perturbed slightly, would dramatically alter the fit




```
slide = ƒ()
```

```
<style>
```

```
hl = ▸Object {highlight: ƒ(e, t, a, c), highlightAuto: ƒ(e, t), fixMarkup: ƒ(e), highlightBlock: ƒ(e), confi
```