Introduction to Regression

Part A - kNN

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai

Lecture Outline

Part A: Statistical Modeling

k-Nearest Neighbors (kNN)

Part B: Model Fitness

How does the model perform predicting?

Part B: Comparison of Two Models

How do we choose from two different models?

Part C: Linear Models

Predicting a Variable

Let's imagine a scenario where we'd like to predict one variable using another (or a set of other) variables.

Examples:

- Predicting the number of views, a **TikTok** video will get next week based on video length, the date it was posted, the previous number of views, etc.
- Predicting which movies, a Netflix user will rate highly based on their previous movie ratings, demographic data, etc.

Working example

The **Advertising data set** consists of the sales of a particular product in 200 different markets, and advertising budgets for the product in each of those markets for three different media: *TV, radio, and newspaper.* Everything is given in units of $1000.

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani "

Response vs. Predictor Variables

There is an asymmetry in many of these problems:

The variable we would like to predict may be more difficult to measure, is more important than the other(s), or maybe directly or indirectly influenced by the other variable(s).

Thus, we'd like to define two categories of variables:

- variables whose values we want to predict
- variables whose values we use to make our prediction

Response vs. Predictor Variables

X
predictors
features
covariates

Y
outcome
response variable
dependent variable

n observations

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

p predictors

Response vs. Predictor Variables

$$X = X_1, \dots, X_p$$
$$X_j = x_{1j}, \dots, x_{ij}, \dots, x_{nj}$$

predictors
features
covariates

$$Y = y_1, \dots, y_n$$

outcome
response variable
dependent variable

n **observations**

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

p **predictors**

Response vs. Predictor Variables

This is called X: a.k.a.
The Design Matrix

y:
The response variable

n **observations**

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

sales
22.1
10.4
9.3
18.5
12.9

Response vs. Predictor Variables

This is called X: a.k.a.
The Design Matrix

y:
The response variable

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

sales
22.1
10.4
9.3
18.5
12.9

Capital letters mean matrices,

Response vs. Predictor Variables

This is called X: a.k.a.
The Design Matrix

y
The response variable

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

sales
22.1
10.4
9.3
18.5
12.9

Capital letters mean matrices, lower case letters mean vectors

Sklearn expects certain dimensions

```
>>> X.shape
(n,m)
```

```
>>> y.shape
(n,) OR (n,1)
```

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

sales
22.1
10.4
9.3
18.5
12.9

Sklearn expects certain dimensions

```
>>> X.shape
(n,) OR (n,1)
```

```
>>> y.shape
(n,) OR (n,1)
```

n observations

TV
230.1
44.5
17.2
151.5
180.8

sales
22.1
10.4
9.3
18.5
12.9

Pre-breakout room Pandas Review

⏸ df[['x']] vs df['x']

What is the difference between the two operations above for a valid dataframe with a column named 'x'.

A. `df[['x']]` returns a `pd.DataFrame` object whereas `df['x']` returns a `pd.Series` object

B. `df[['x']]` returns a `pd.Series` object whereas `df['x']` returns a `pd.DataFrame` object

C. `df[['x']]` is an invalid operation

D. `df['x']` is an invalid operation

Statistical Model

True vs. Statistical Model

We will assume that the response variable, Y, relates to the predictors, X, through some unknown function expressed generally as:

$$Y = f(X) + \varepsilon$$

Here, f is the unknown function expressing an underlying rule for relating Y to X, ε is the random amount (unrelated to X) that Y differs from the rule $f(X)$.

A ***statistical model*** is any algorithm that estimates f. We denote the estimated function as $\widehat{f}$.

Prediction vs. Estimation

For some problems, what's important is obtaining $\hat{f}$, our estimate of f. These are called ***inference*** problems.

When we use a set of measurements, $(x_{i,1}, \dots, x_{i,p})$ to predict a value for the response variable, we denote the ***predicted*** value by:

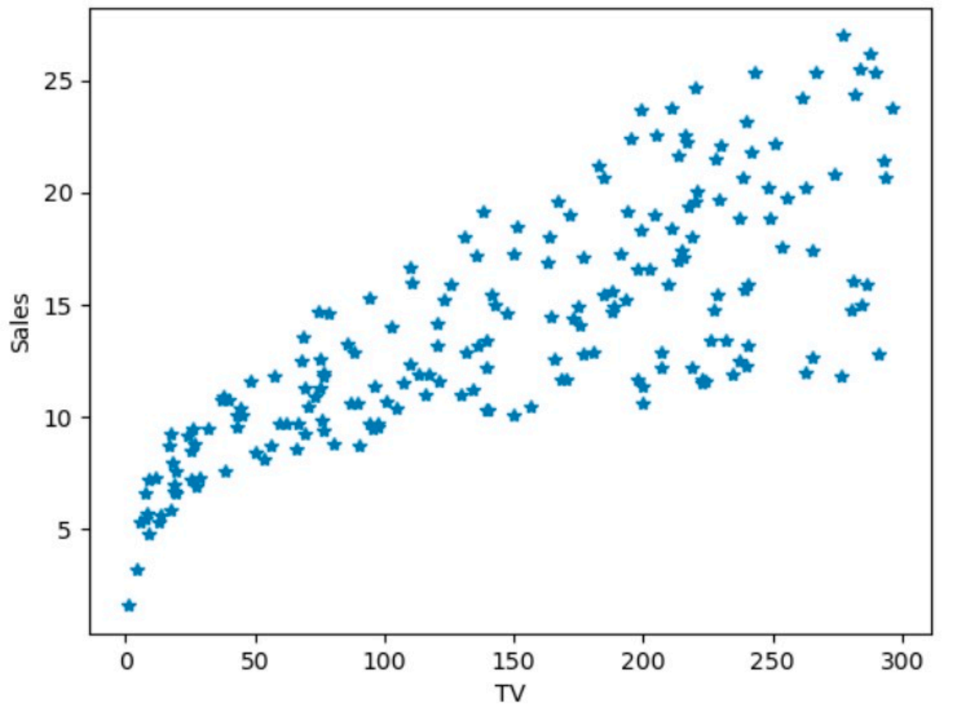
$$\hat{y}_i = \hat{f}(x_{i,1}, \dots, x_{i,p}).$$

For some problems, we don't care about the specific form of $\hat{f}$, we just want to make our predictions $\hat{y}$'s as close to the observed values y's as possible. These are called ***prediction problems***.

Example: predicting sales

Motivation: Predict Sales

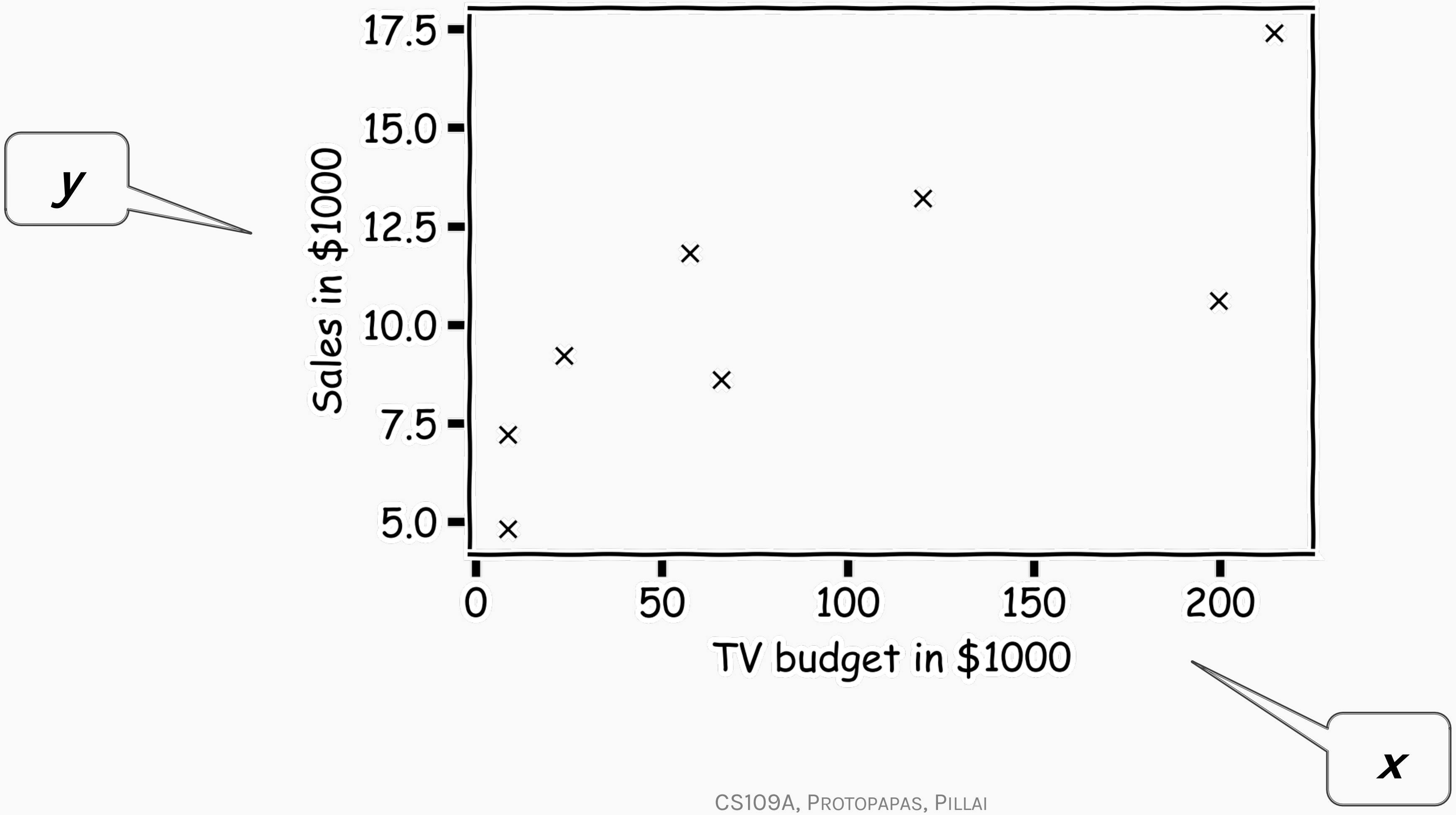
Build a model to **predict** sales based on TV budget



The response, y, is the sales

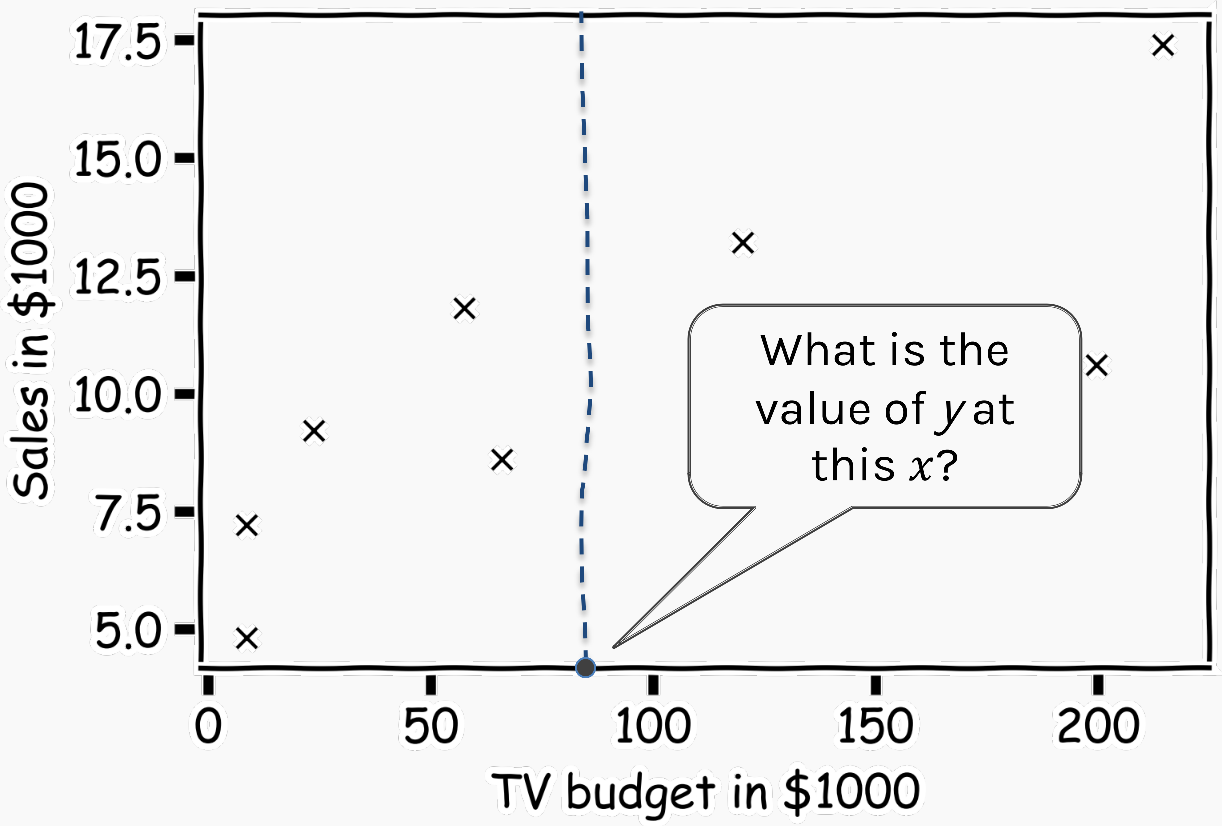
The predictor, x, is TV budget

Statistical Model



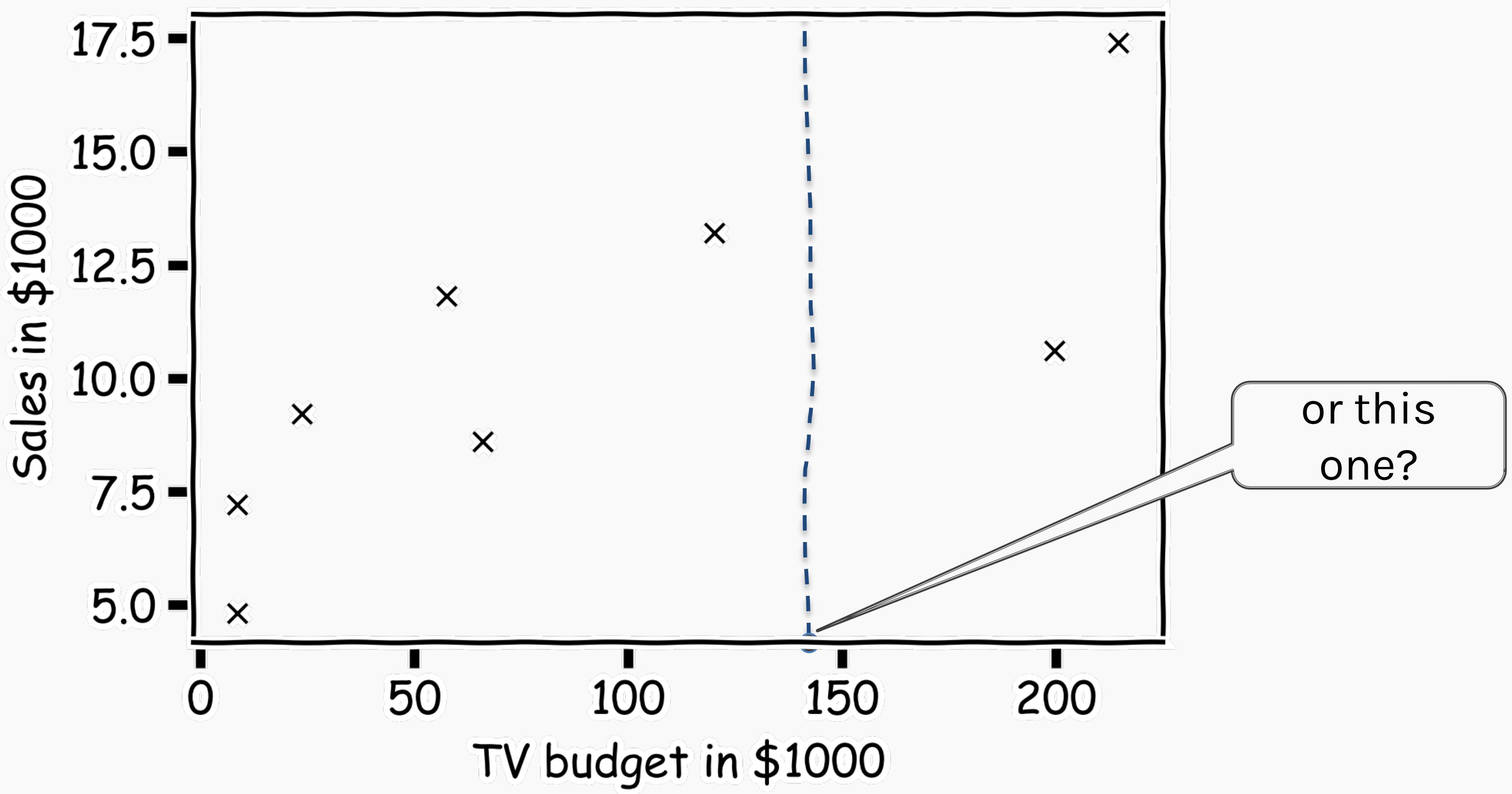
Statistical Model

How do we predict y for some x?



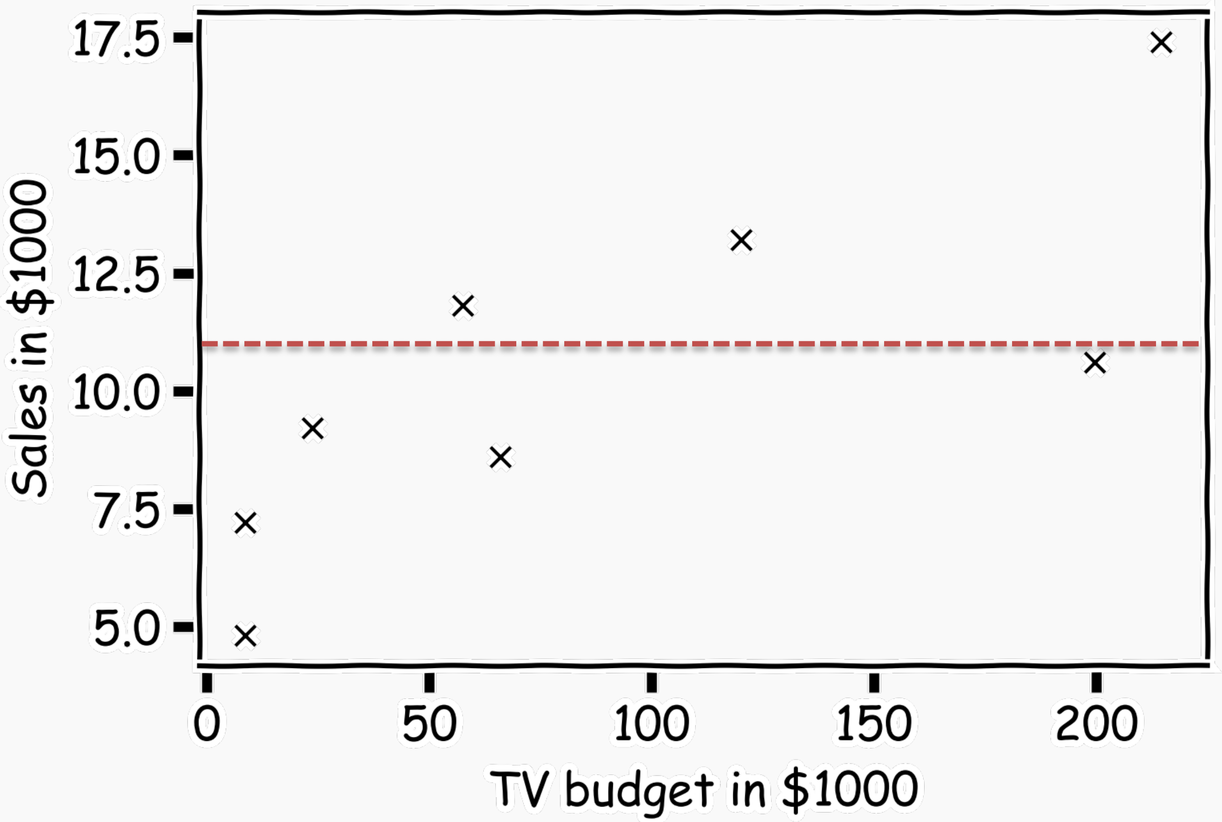
Statistical Model

How do we predict y for some x?



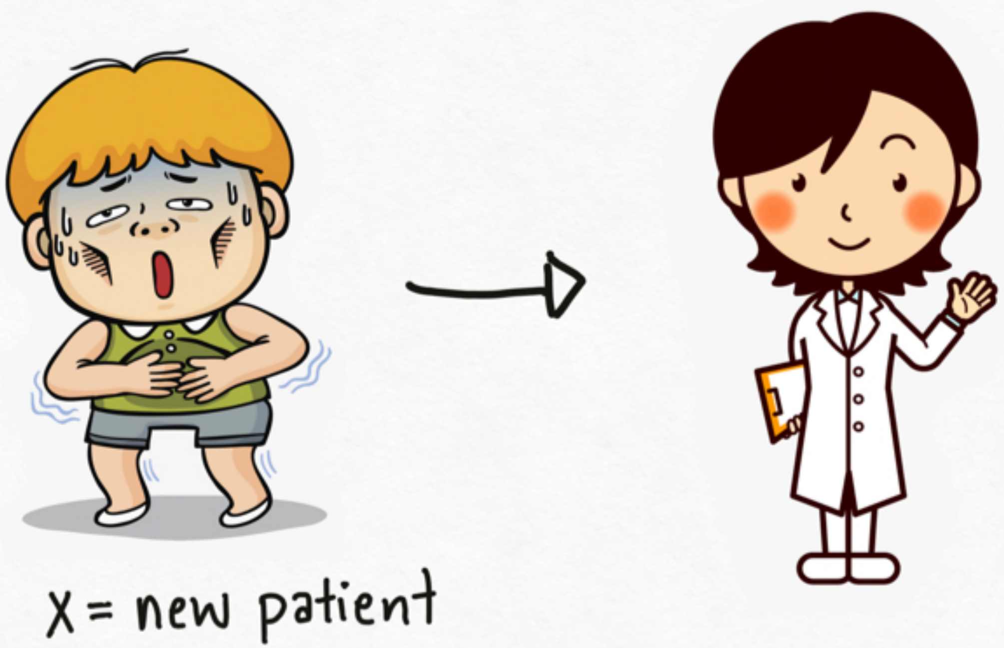
Statistical Model

A simple idea is to take the mean of all y's: $\frac{1}{n}\sum_{i=1}^{n} y_i$

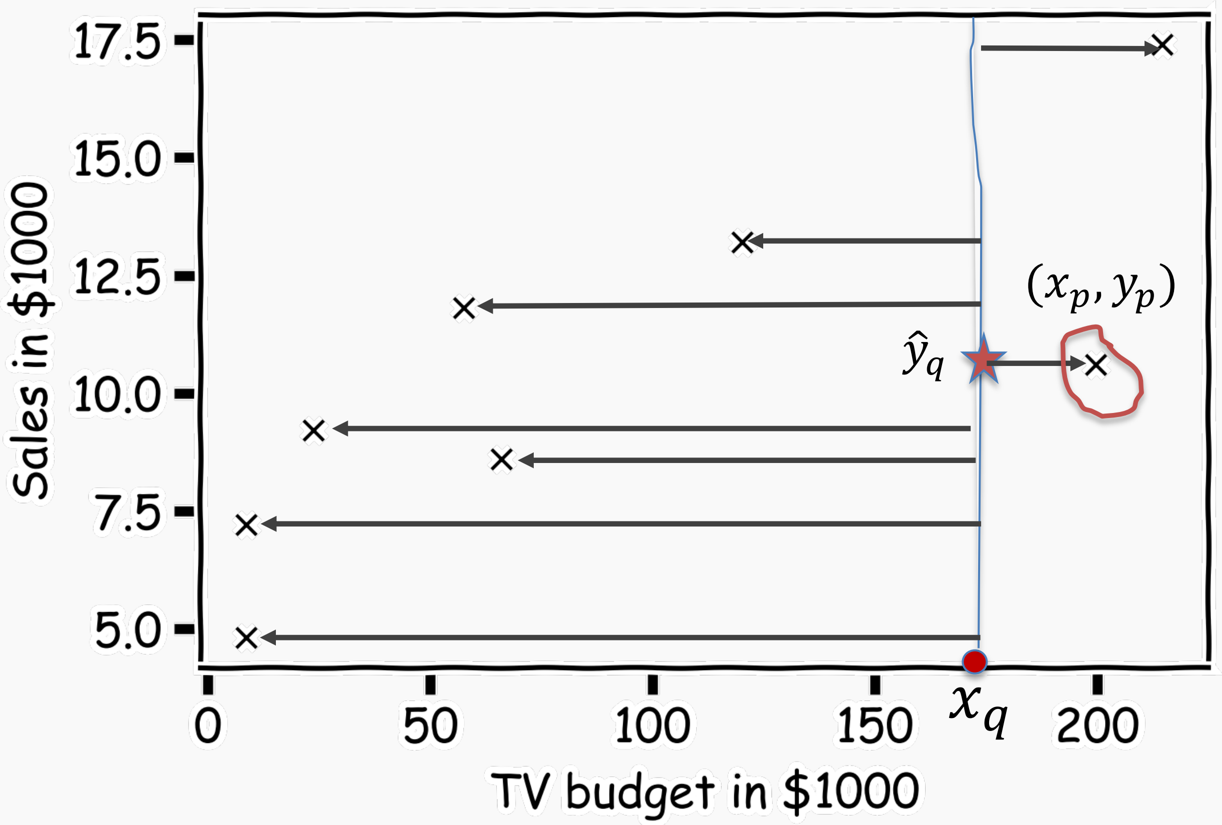


$$\frac{1}{n}\sum_{i=1}^{n} y_i$$

k-Nearest Neighbors – kNN



Simple Prediction Model



What is $\hat{y}_q$ at some x_q ?

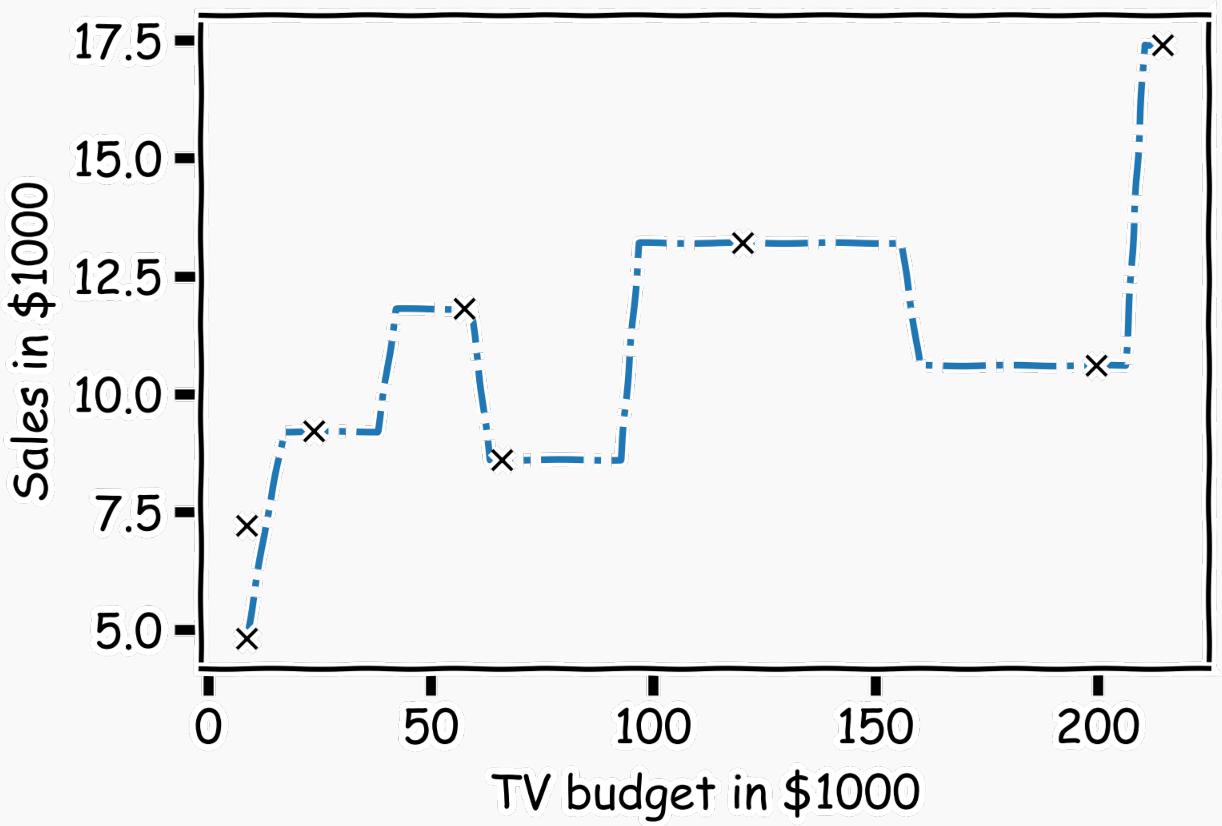
Find distances to all other points $D(x_q, x_i)$

Find the nearest neighbor, (x_p, y_p)

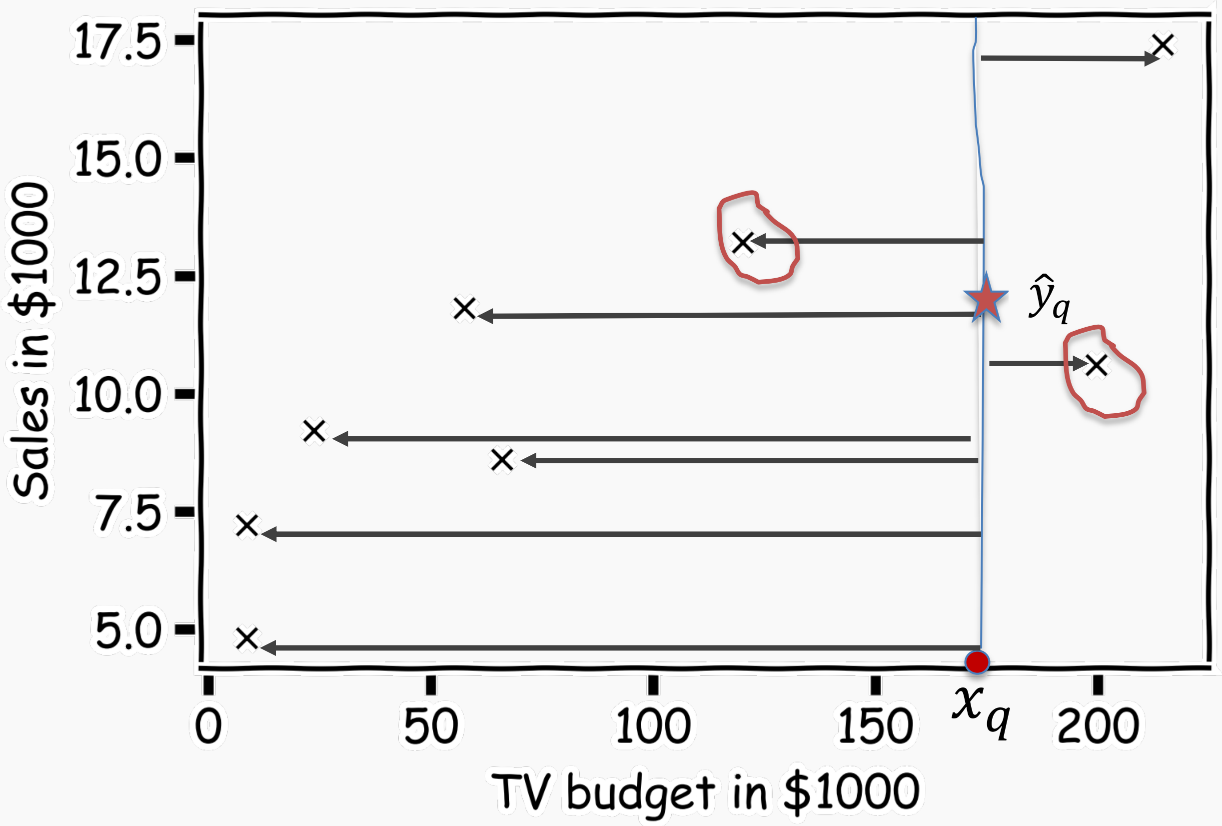
Predict $\hat{y}_q = y_p$

Simple Prediction Model

Do the same for "all" $x's$



Extend the Prediction Model



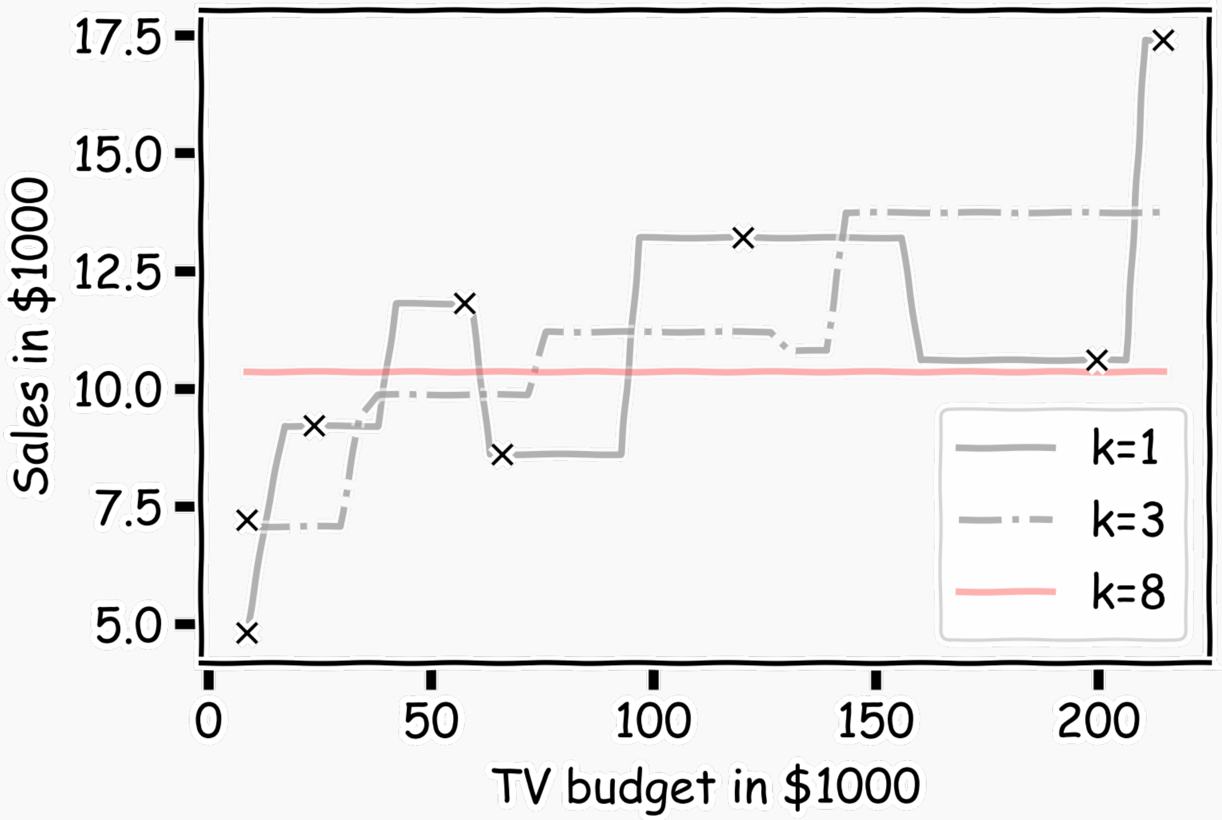
What is $\hat{y}_q$ at some x_q ?

Find distances to all other points $D(x_q, x_i)$

Find the k-nearest neighbors, $x_{q_1}, \ldots, x_{q_k}$

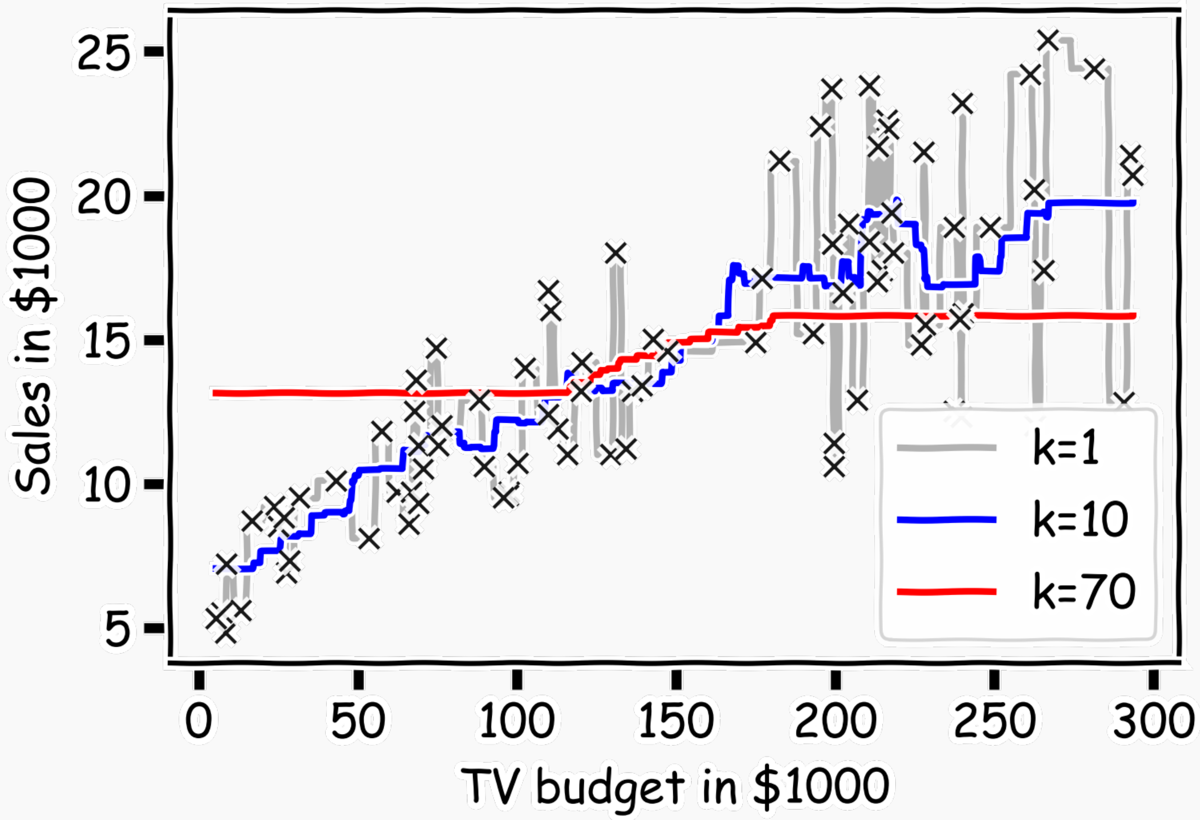
Predict $\hat{y}_q = \frac{1}{k}\sum_i^k y_{q_i}$

Simple Prediction Models



Simple Prediction Models

We can try different k-models on more data



k-Nearest Neighbors – kNN

The very human way of decision making by similar examples. kNN is a **non-parametric** learning algorithm.

The k-Nearest Neighbor Algorithm:

Given a dataset $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$. For every new X:

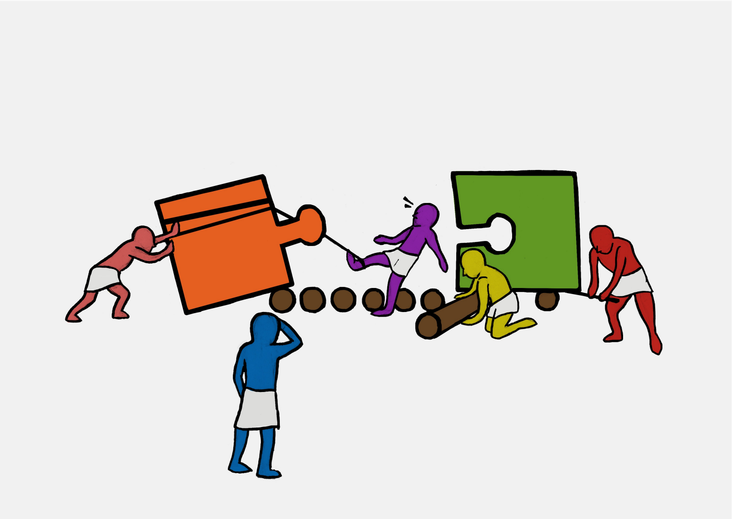
1. Find the k-number of observations in D most similar to X:

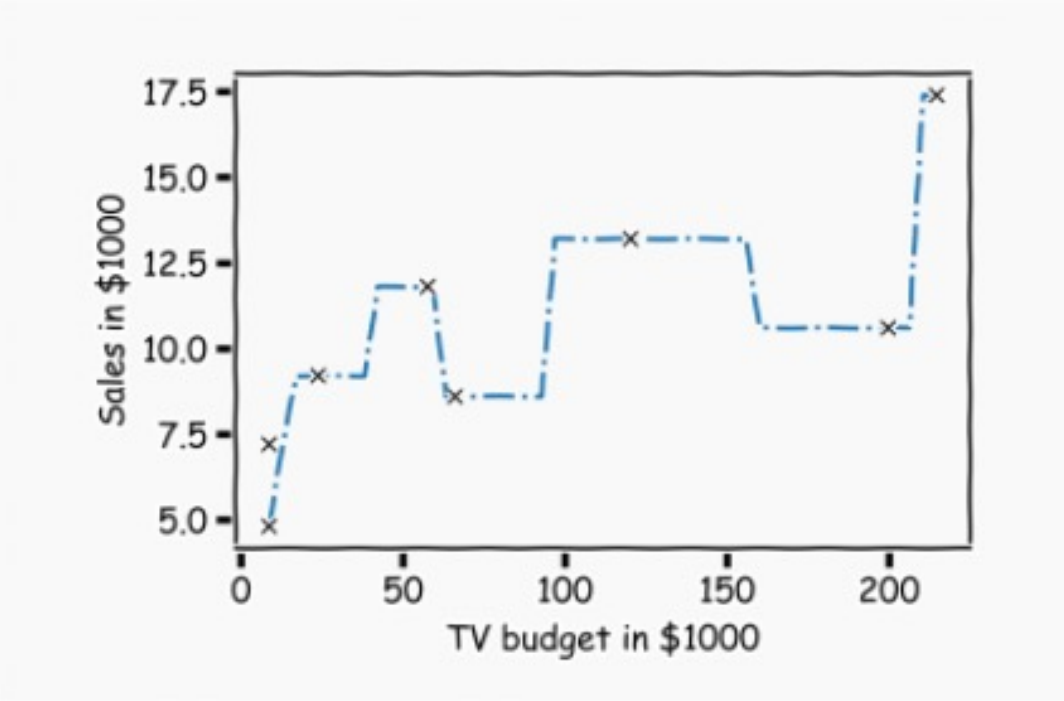
$$\{(x^{(n_1)}, y^{(n_1)}), \ldots, (x^{(n_k)}, y^{(n_k)})\}$$

   These are called the k-nearest neighbors of x

2. Average the output of the k-nearest neighbors of x

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} y^{(n_k)}$$





👨‍🏫 Demonstrate Ex Simple data plotting

🏋️‍♀️ Simple kNN Regression by hand