Patrice Koehl

Supervised Learning: k-Nearest Neighbors

Predicting

Let’s imagine a scenario where we would like to predict the value of one variable using another (or a set of other) variables.

Examples:

- Predicting the effect of a medication based on symptoms experienced by the patient (temperature, pain, some blood results,…)
- Predicting which movies a Netflix user will rate highly based on their previous movie ratings, demographic data, etc.

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

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, may be more important than the other(s), or are probably 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

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

n observations

p predictors

Response vs Predictor Variables

$X_1, X_2, \ldots, X_p$
predictors
features
covariates

$Y_1, Y_2, \ldots, Y_m$
outcome
response variable
dependent variable

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

n observations

p predictors

Statistical Model

We 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 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 $\hat{f}$.

Prediction vs Estimation

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

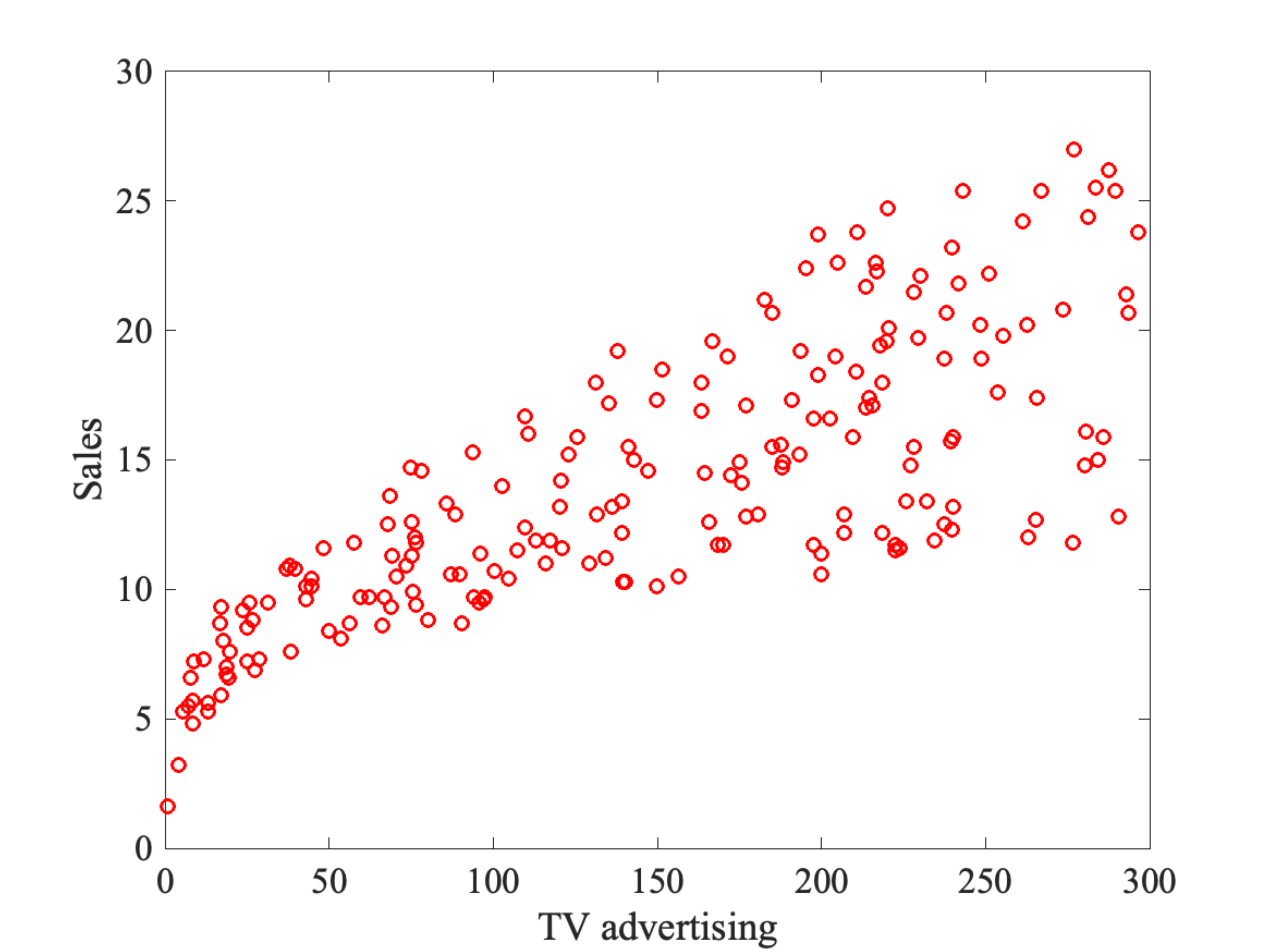
When we use a set of measurements, $(x_{i,1}, \ldots, x_{i,p})$ to predict a value for the response variable, we denote the ***predicted*** value by:

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

For some problems, we do not care about the specific expression 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***.

Prediction Model

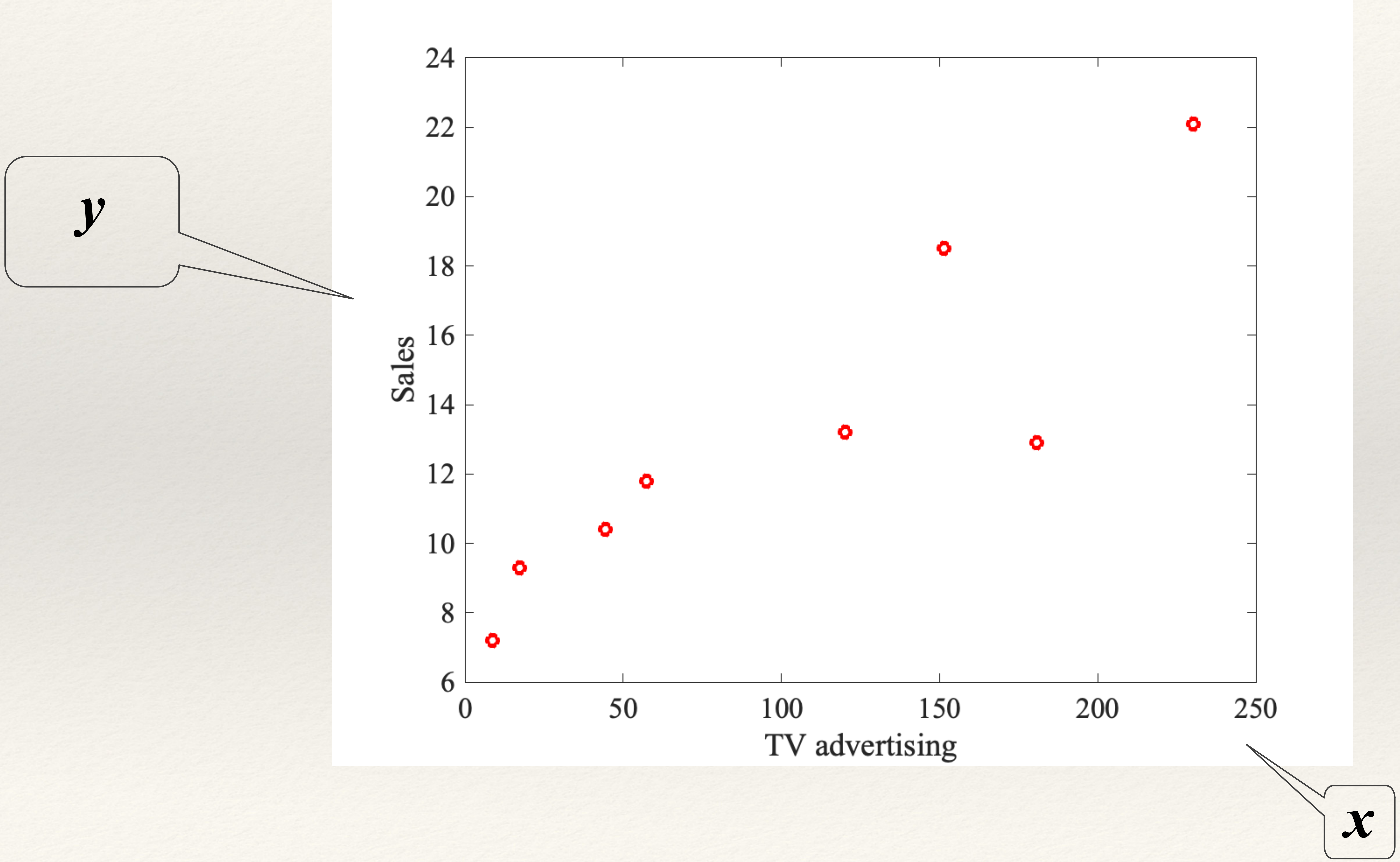
Build a model to **predict** sales based on TV budget



The response, y, is the sales

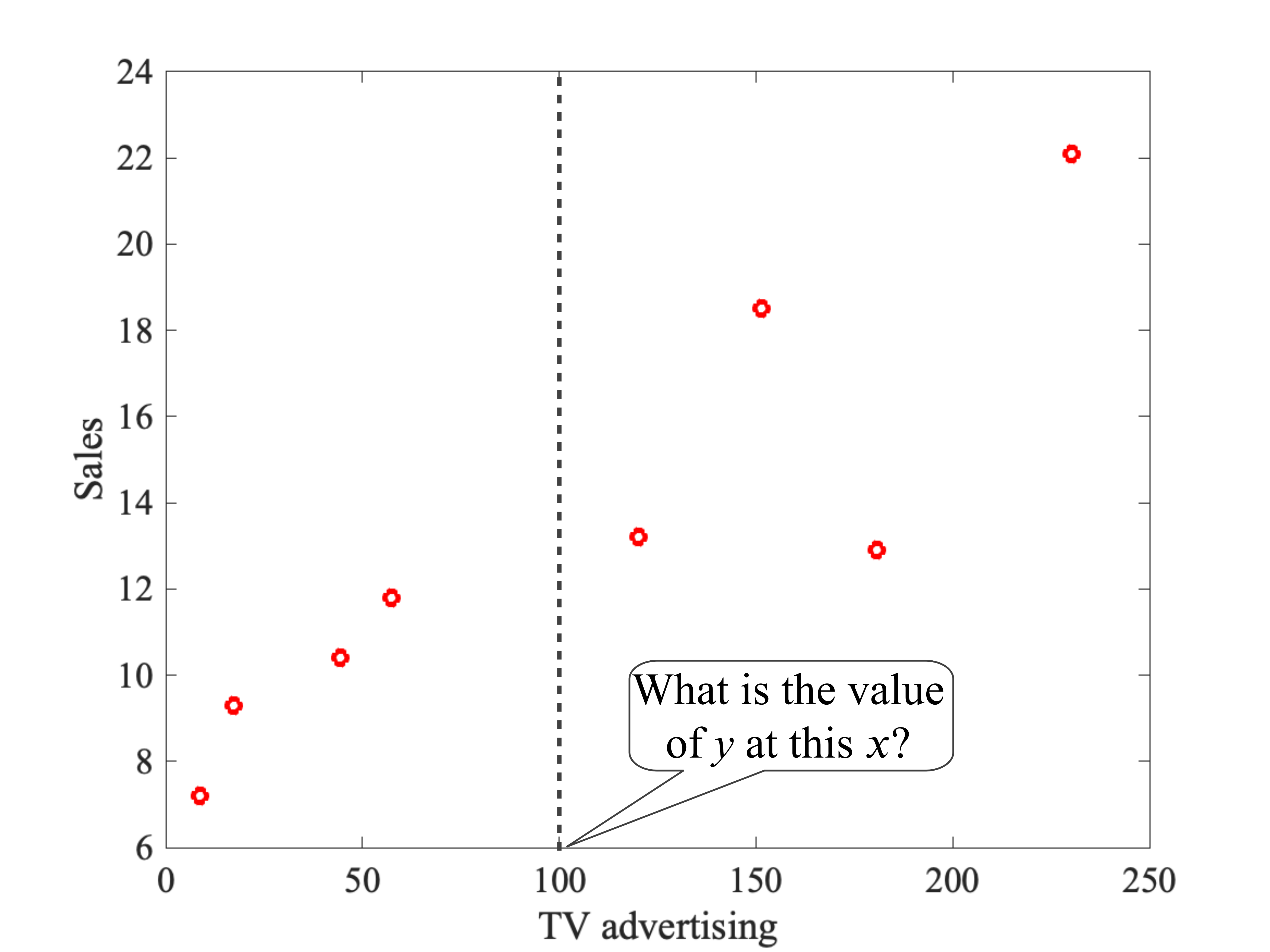
The predictor, x, is TV budget

Prediction Model



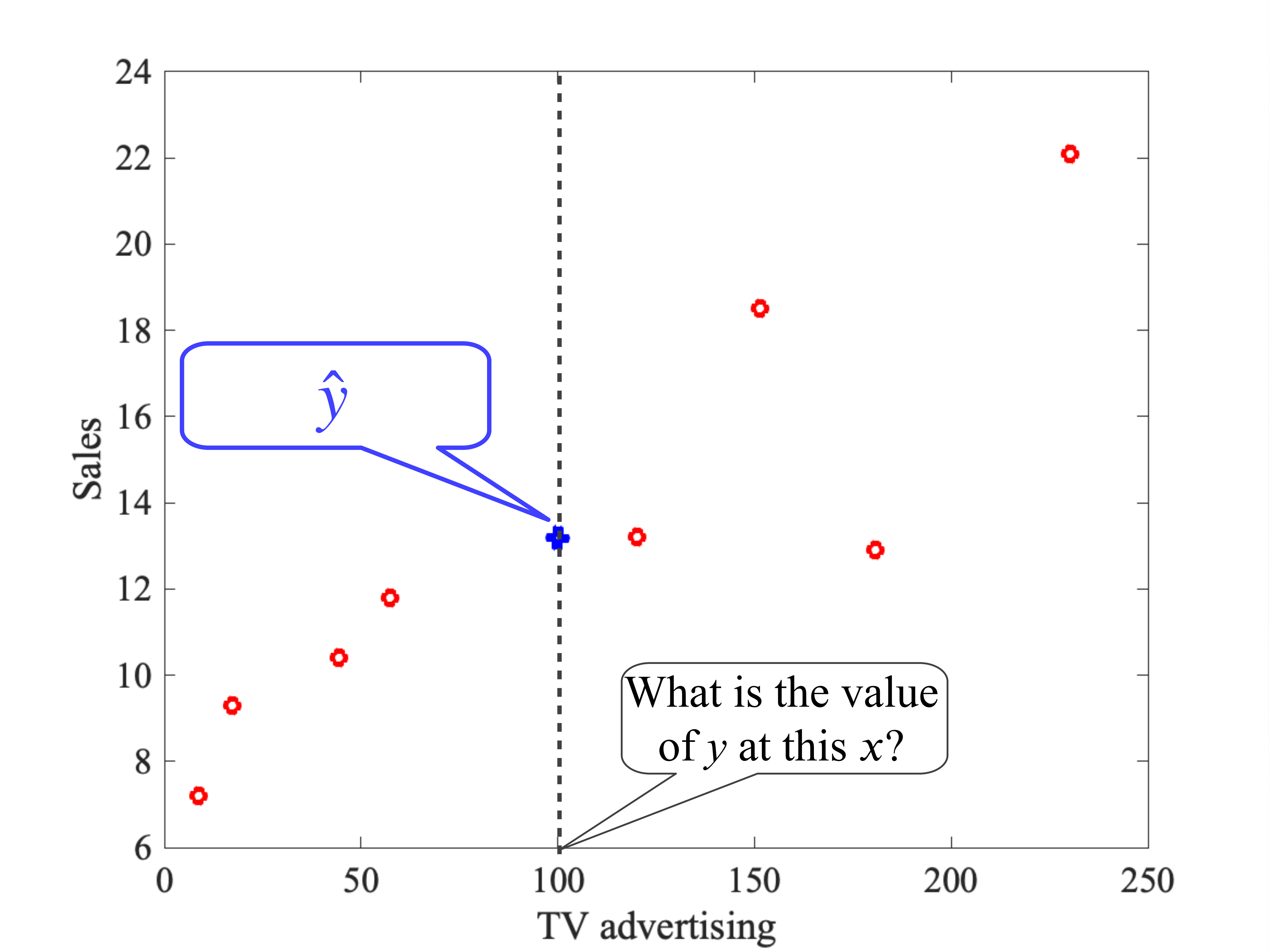
Prediction Model

How do we predict the sales value (y) for a given TV advertising value (x)?

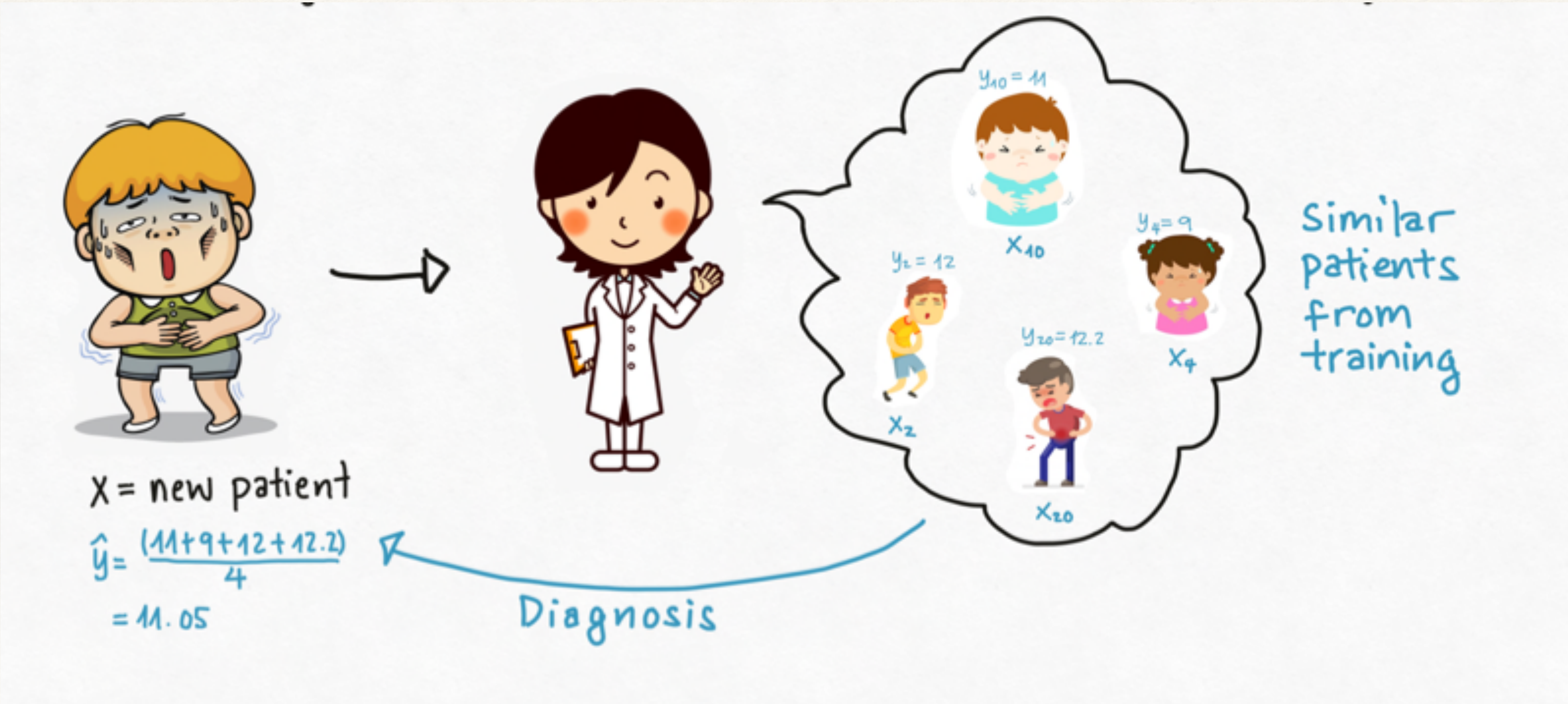


Prediction Model: Average

Simplest idea: table the mean of the existing values: $\hat{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$



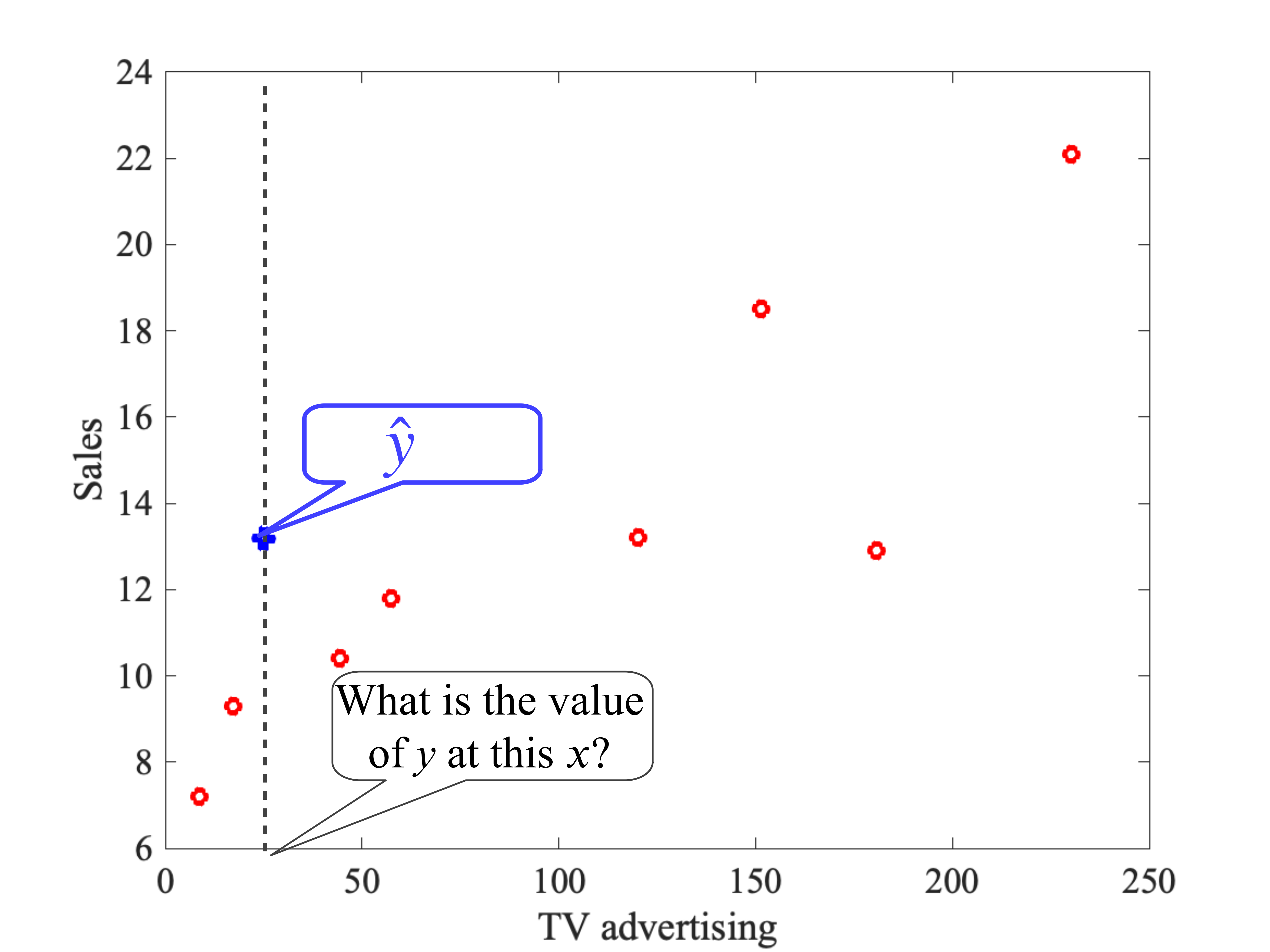
Statistical Model



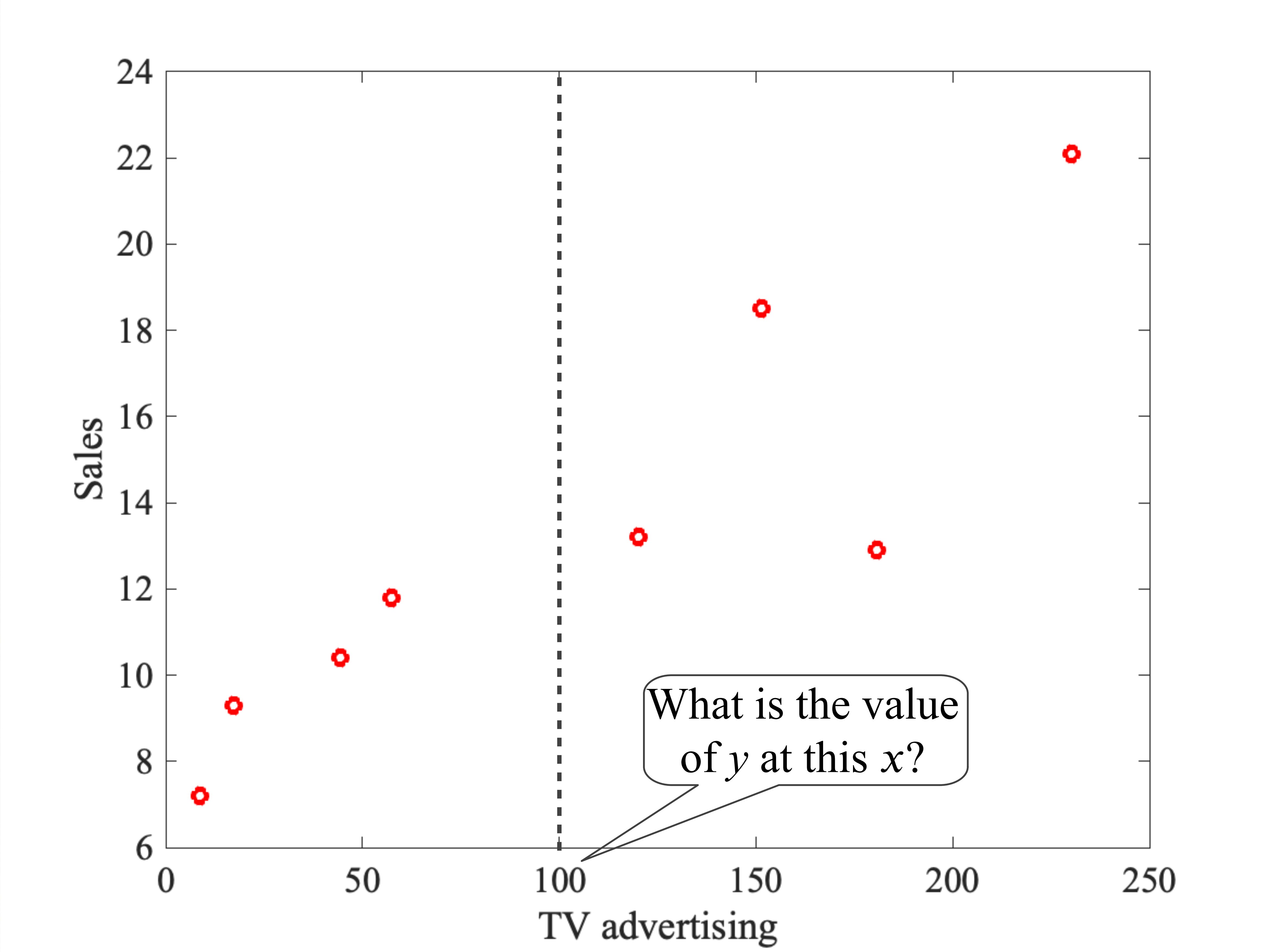
Prediction Model: Average

Simplest idea: table the mean of the existing values: $\hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

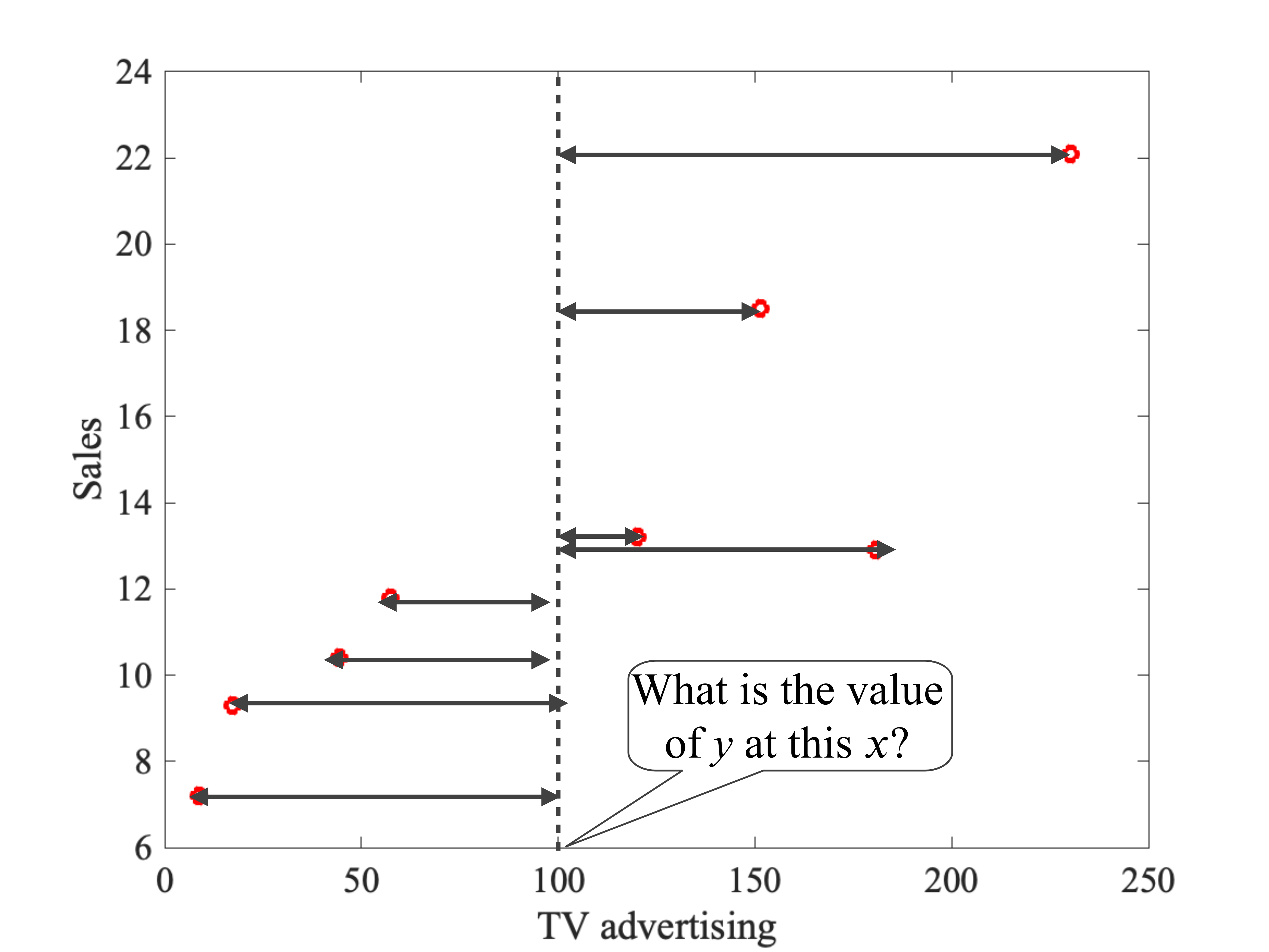
Not always the "best" solution!



Prediction Model: 1-Neighbour

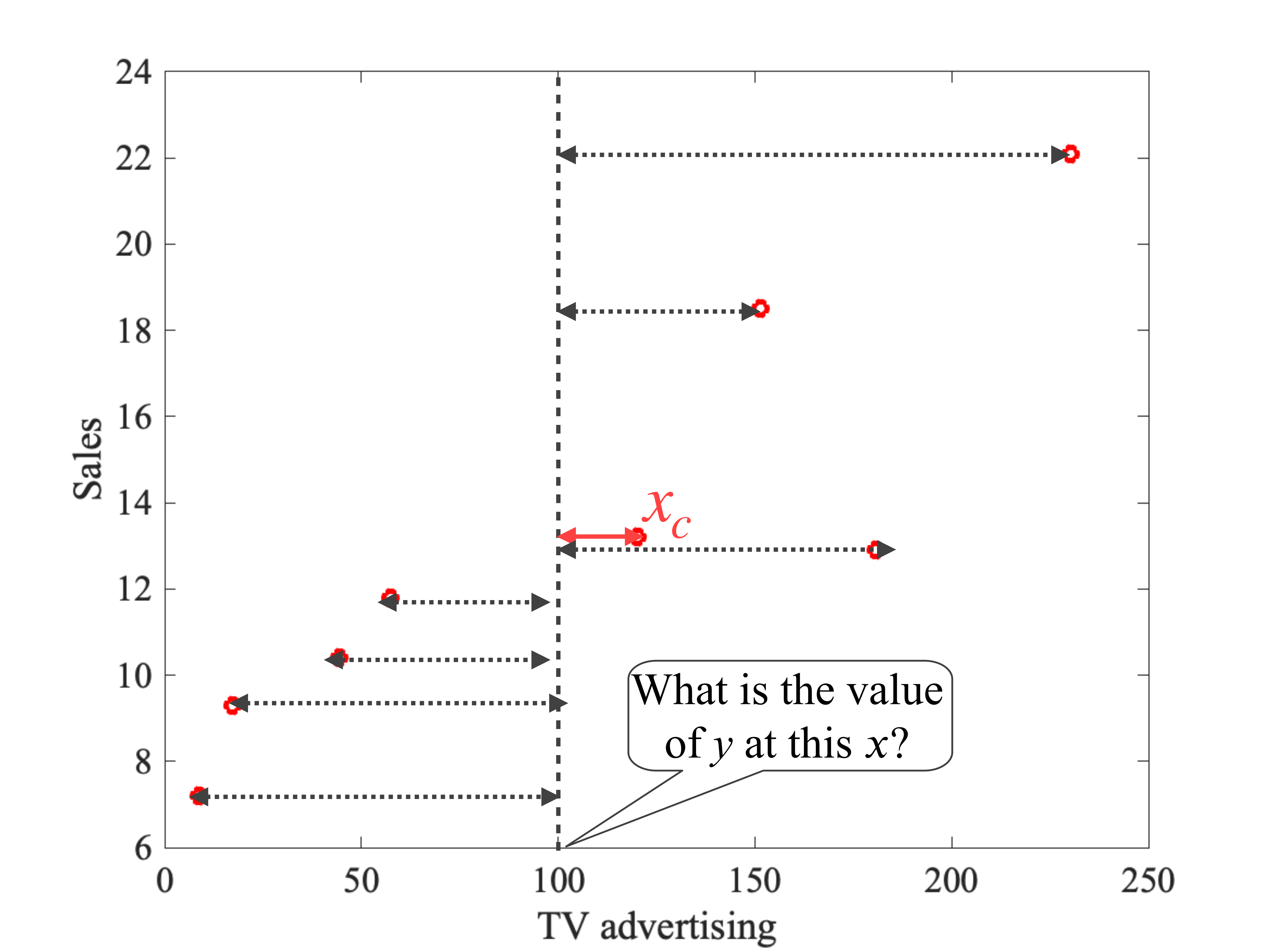


Prediction Model: 1-Neighbour



1. Find distance D(x,x_i) to all other points

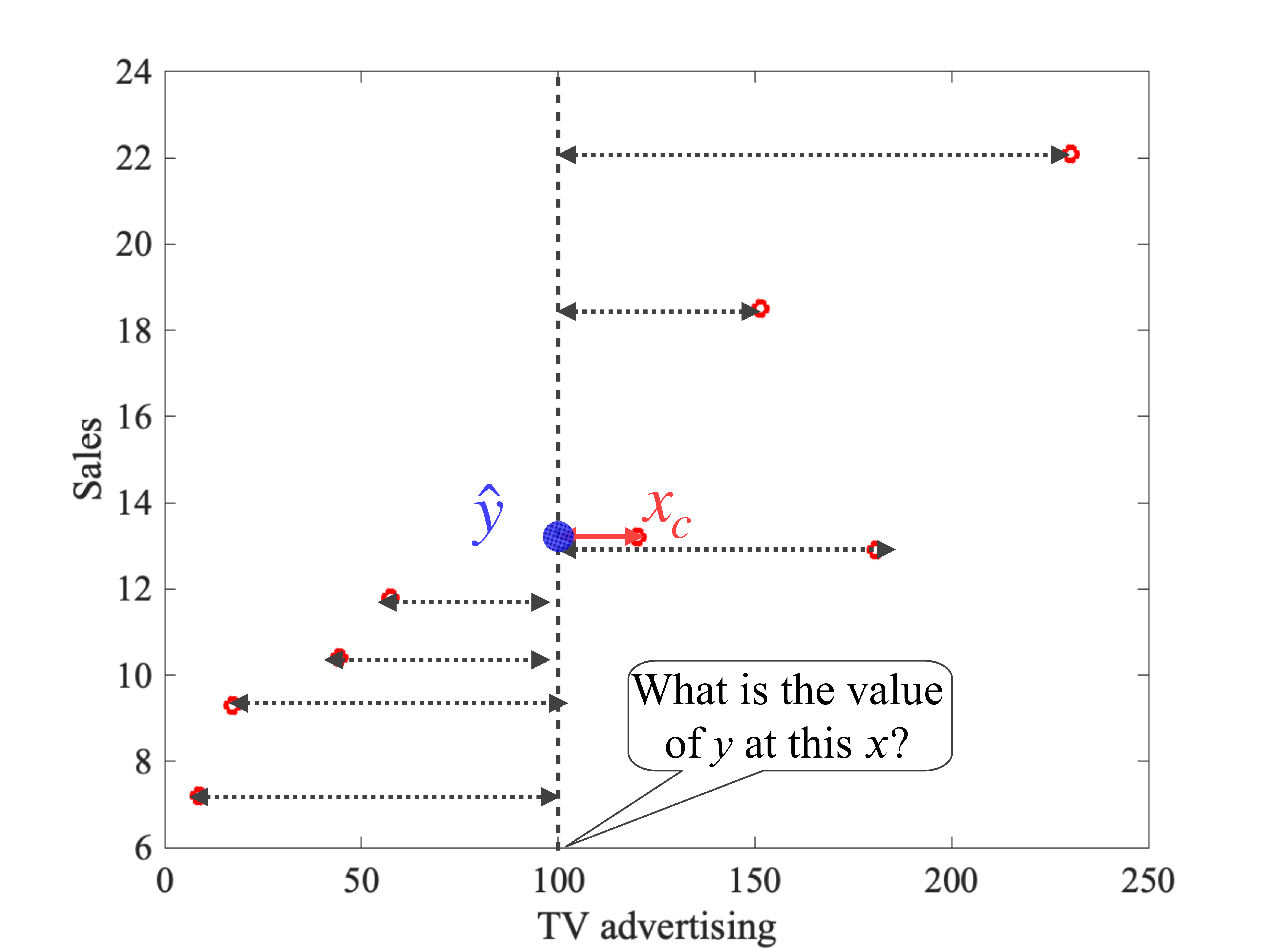
Prediction Model: 1-Neighbour



1. Find distance D(x,x_i) to all other points

2. Find closest x_c

Prediction Model: 1-Neighbour



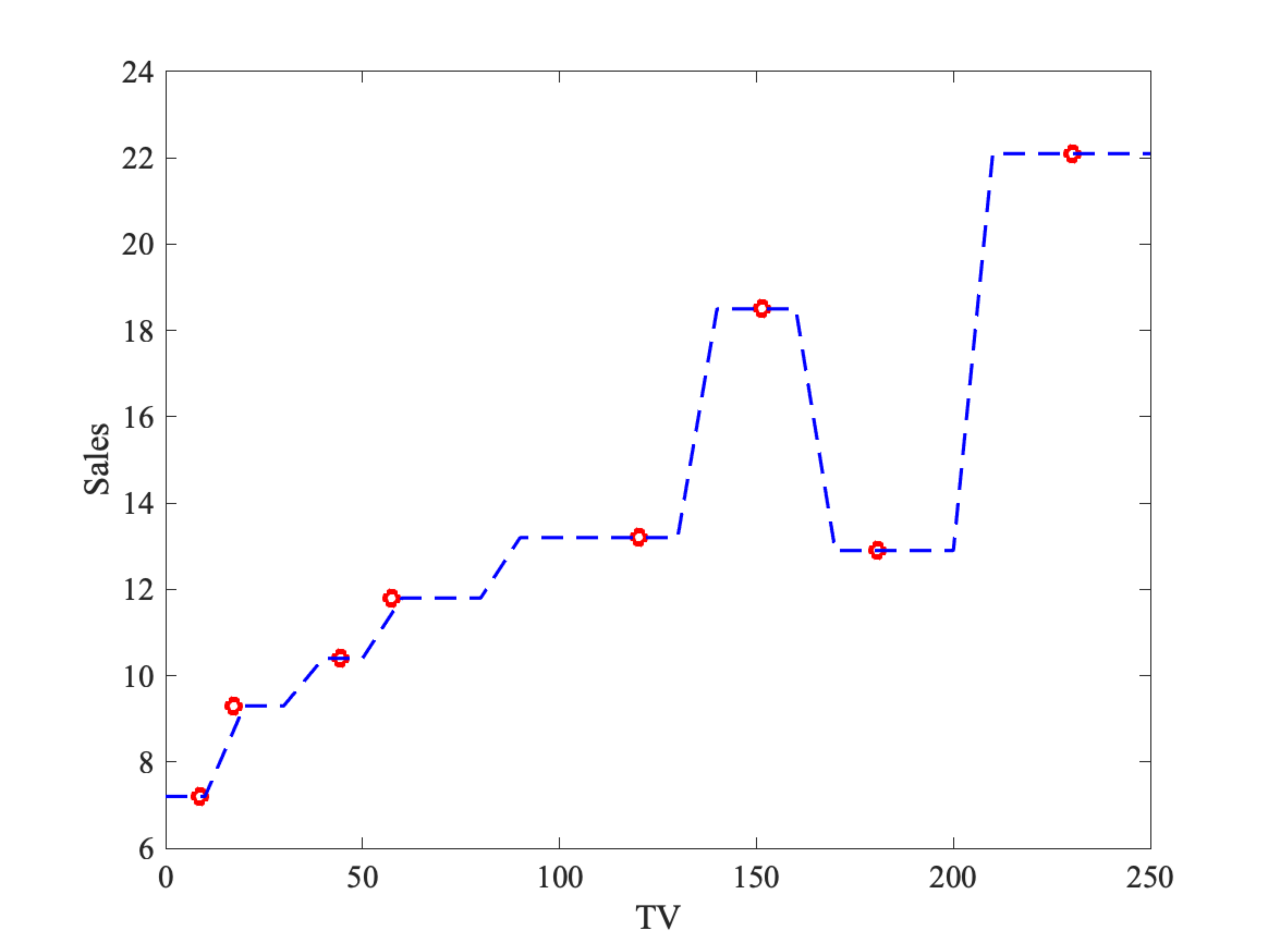
1. Find distance D(x,x_i) to all other points

2. Find closest x_c

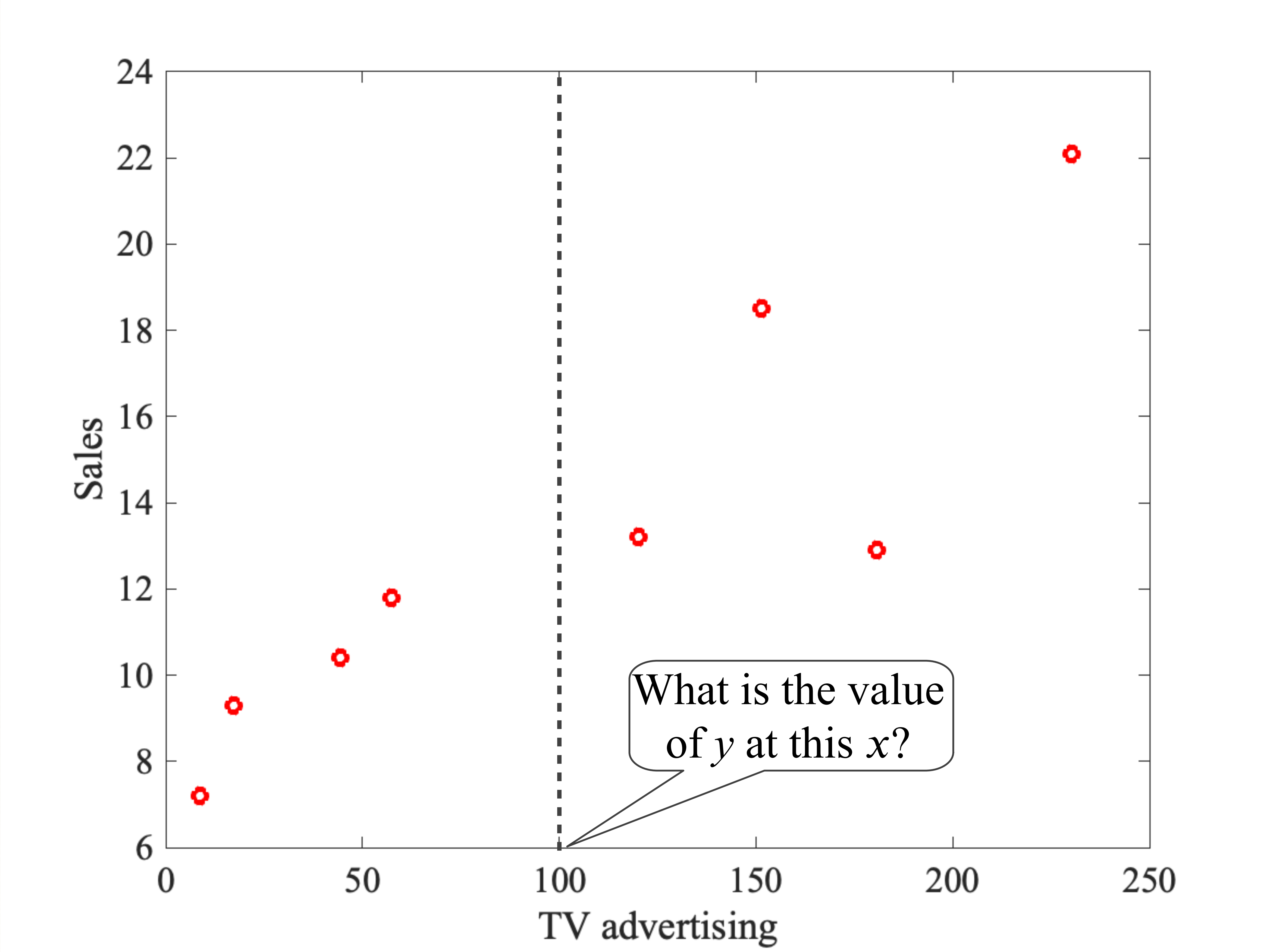
3. Define $\hat{y}(x) = y(x_c)$

Prediction Model: 1-Neighbour

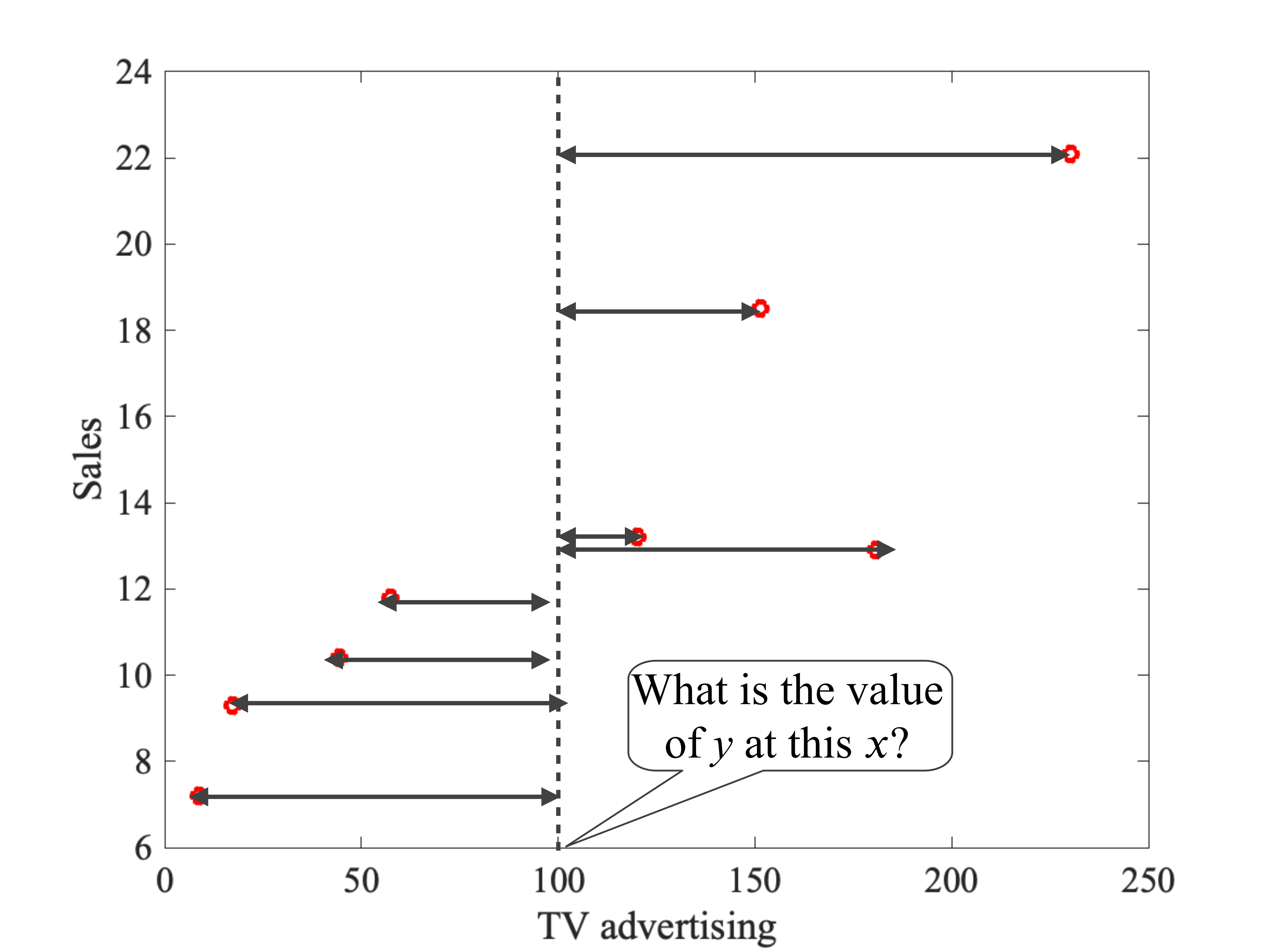
Repeat for all values of x in the range of "TV": this builds a model for y!



Prediction Model: k-Nearest Neighbours

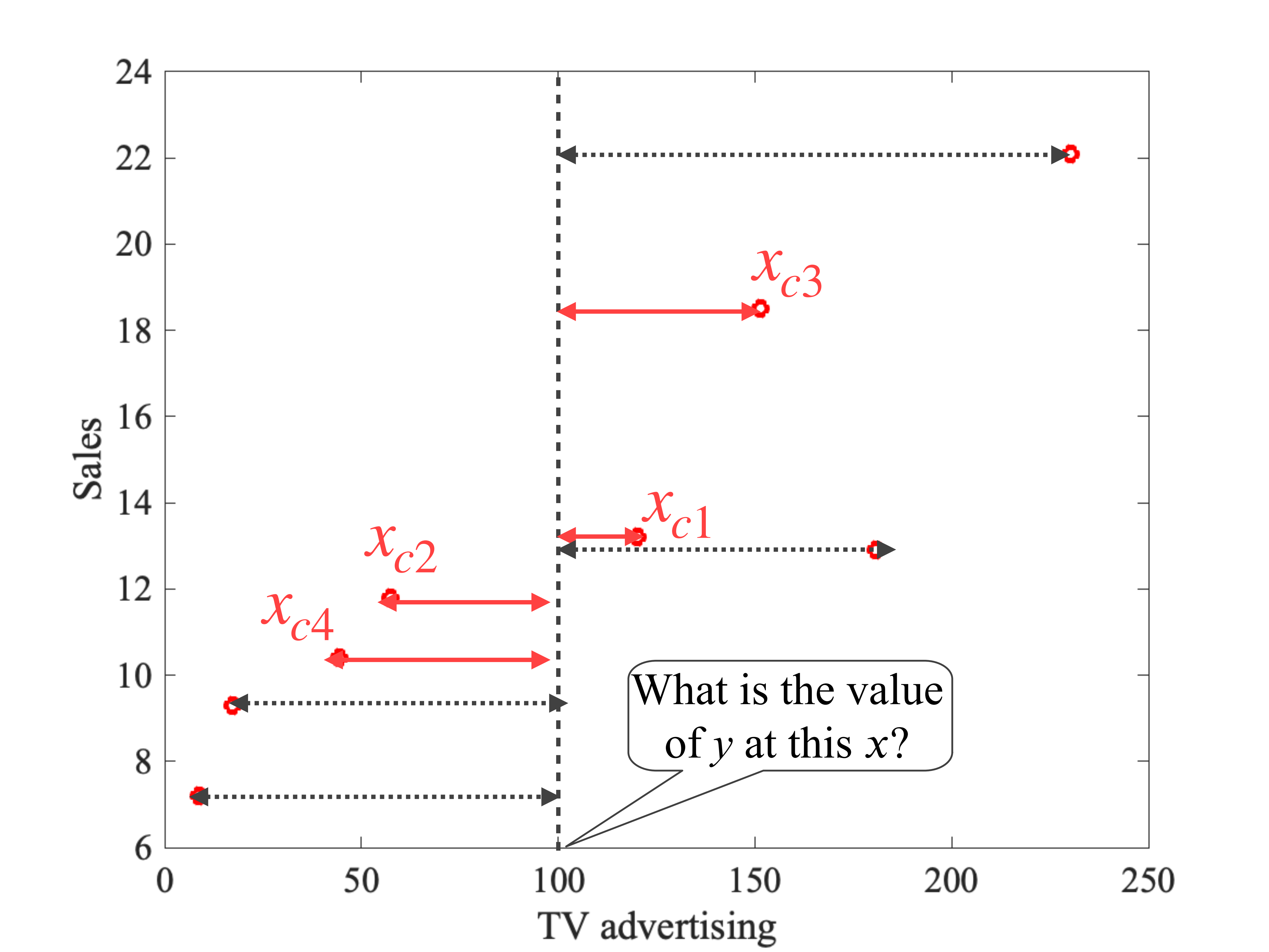


Prediction Model: k-Nearest Neighbours



1. Find distance D(x,x_i) to all other points

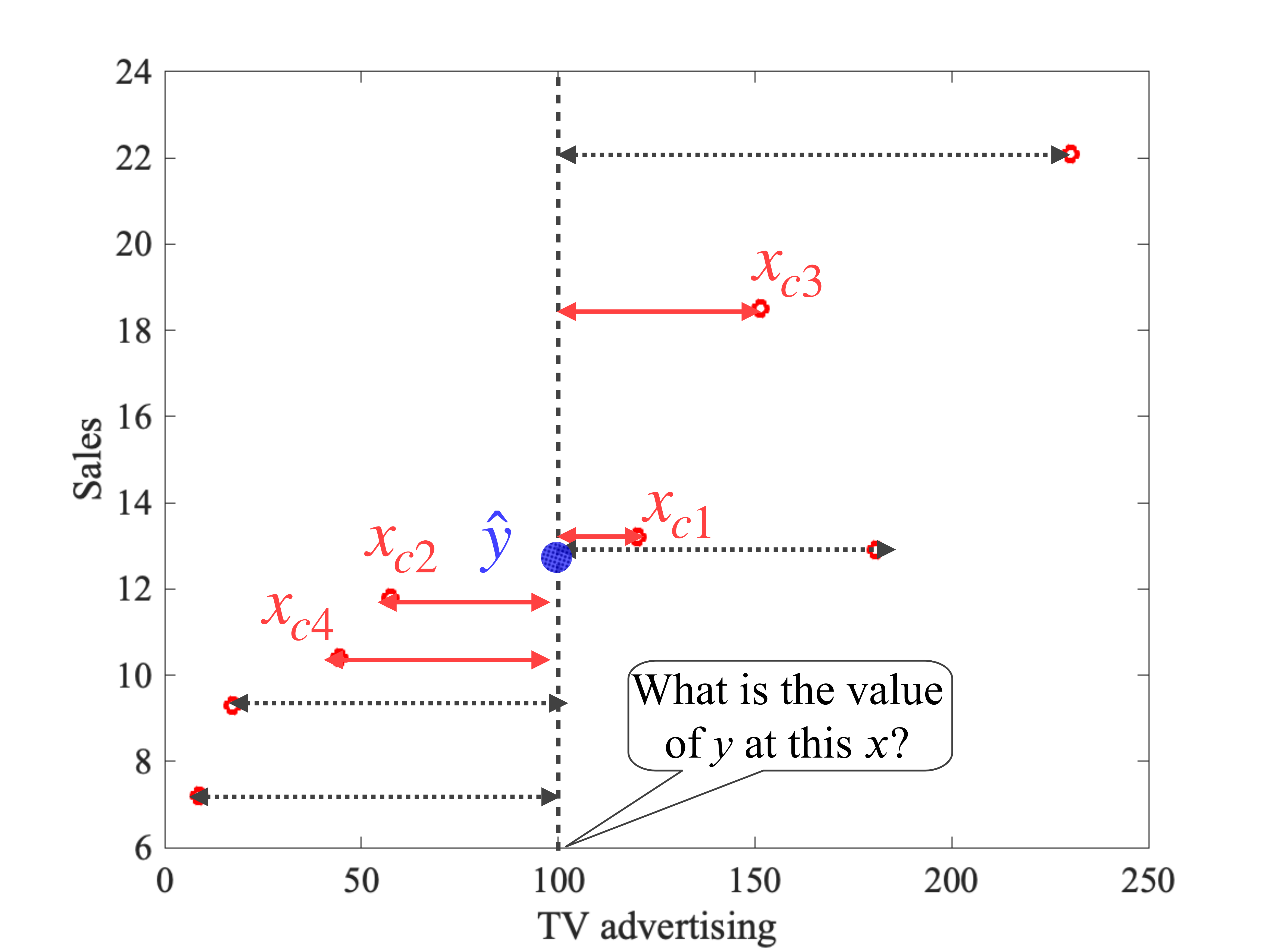
Prediction Model: k-Nearest Neighbours



1. Find distance D(x,x_i) to all other points

2. Find k closest x_{c1}, x_{c2}, x_{c3}, x_{c4} …
   (*here* $k = 4$)

Prediction Model: k-Nearest Neighbours



1. Find distance D(x,x_i) to all other points

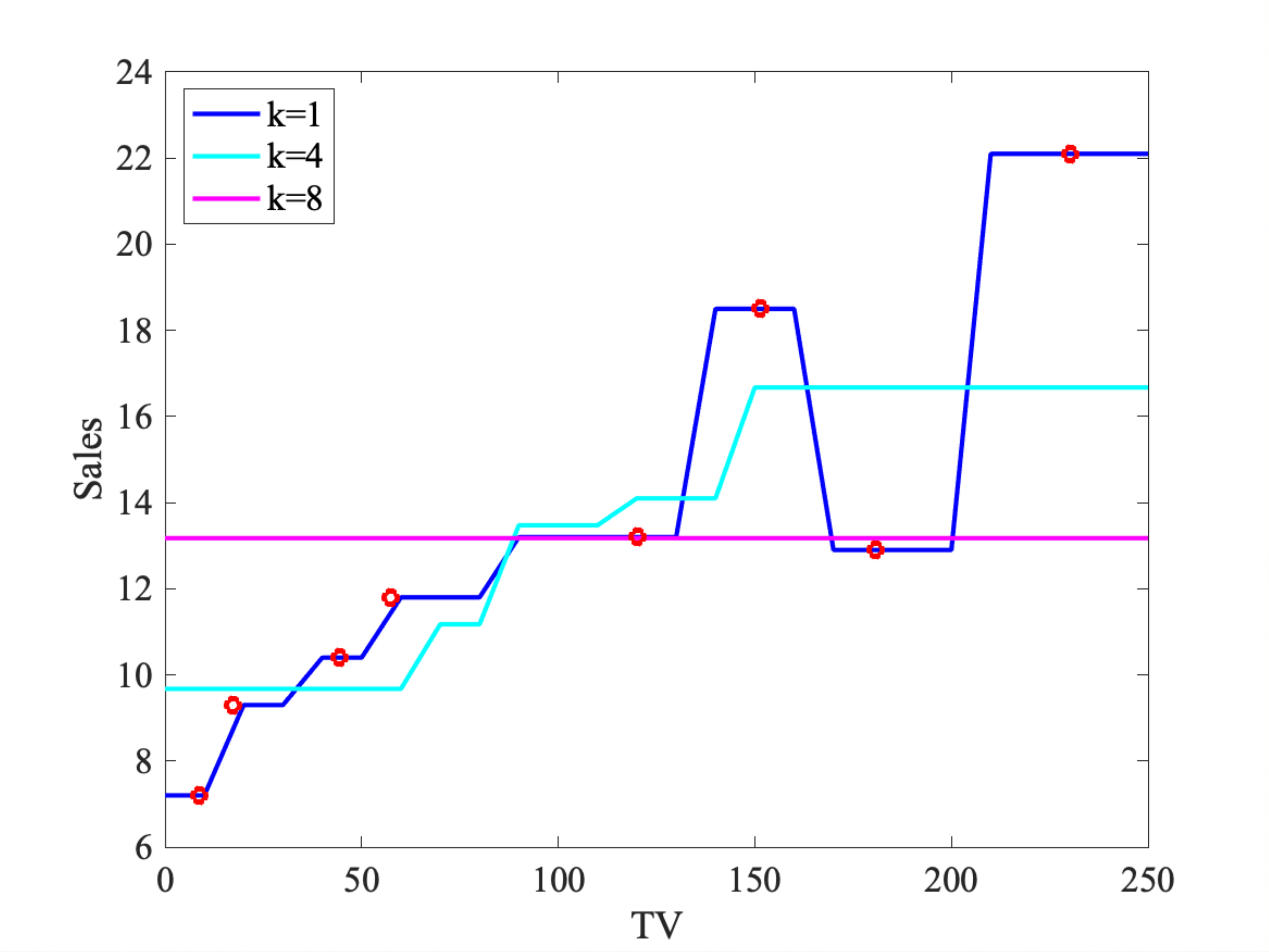
2. Find k closest x_{c1}, x_{c2}, x_{c3}, x_{c4} …
   (*here* $k = 4$)

3. Average y over k-nearest Neighbors

$$\hat{y} = \frac{1}{k} \sum_{j=1}^{k} y(x_{cj})$$

Prediction Model: k-Nearest Neighbours

Repeat for all values of x in the range of “TV” for different k values: this builds different models for y!



Prediction Model: k-Nearest Neighbours

The ***k-Nearest Neighbor (kNN) model*** is an intuitive way to predict a quantitative response variable:

to predict a response for a set of observed predictor values, we use the responses of other observations most similar to it

kNN is a non-parametric learning algorithm. When we say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution.

Note: this strategy can also be applied to classification problems to predict a categorical variable. We will encounter kNN in the lab.

Prediction Model: k-Nearest Neighbours

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:

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

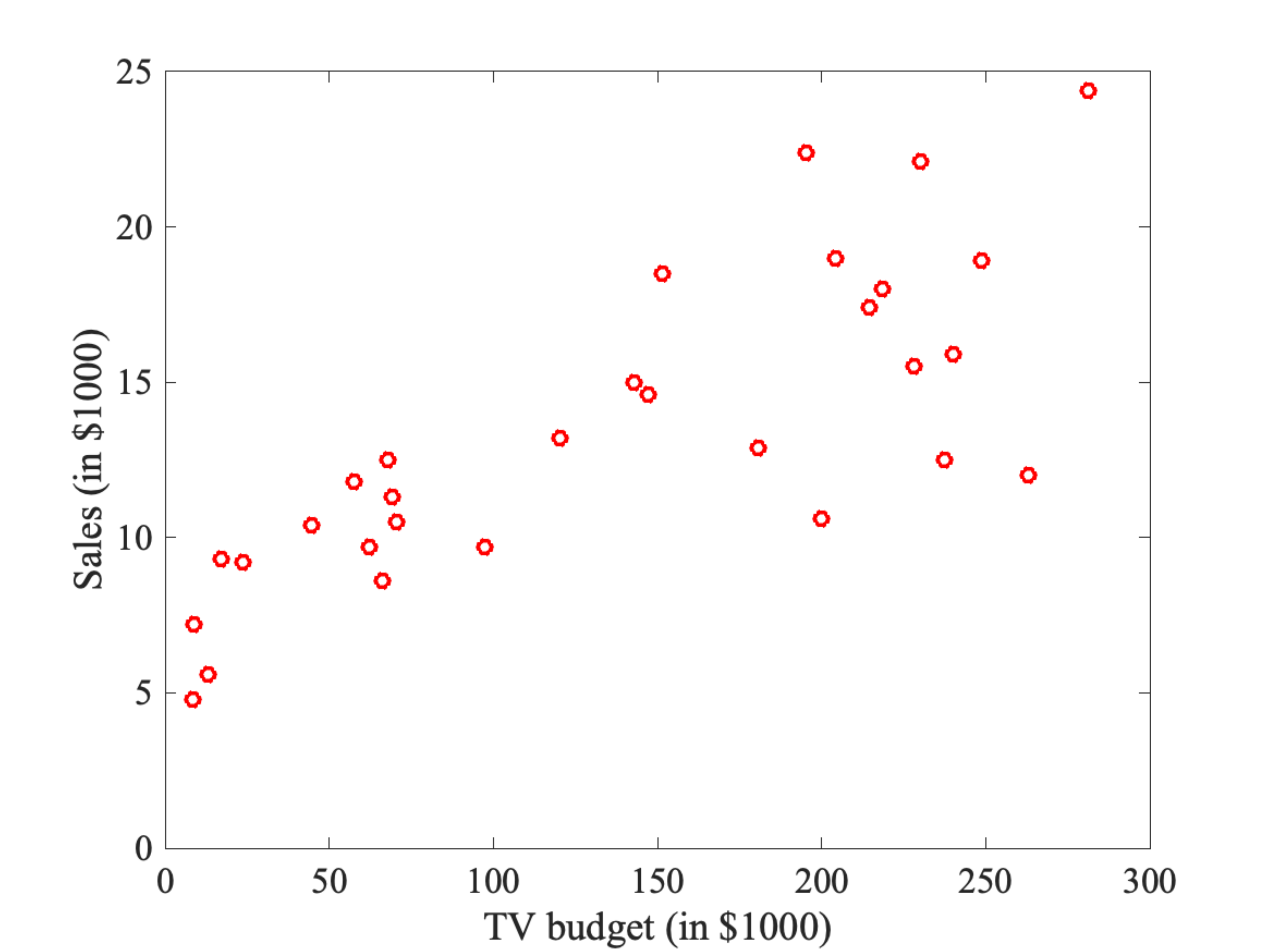
   These are called the k-nearest neighbors of x

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

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y^{n_i}$$

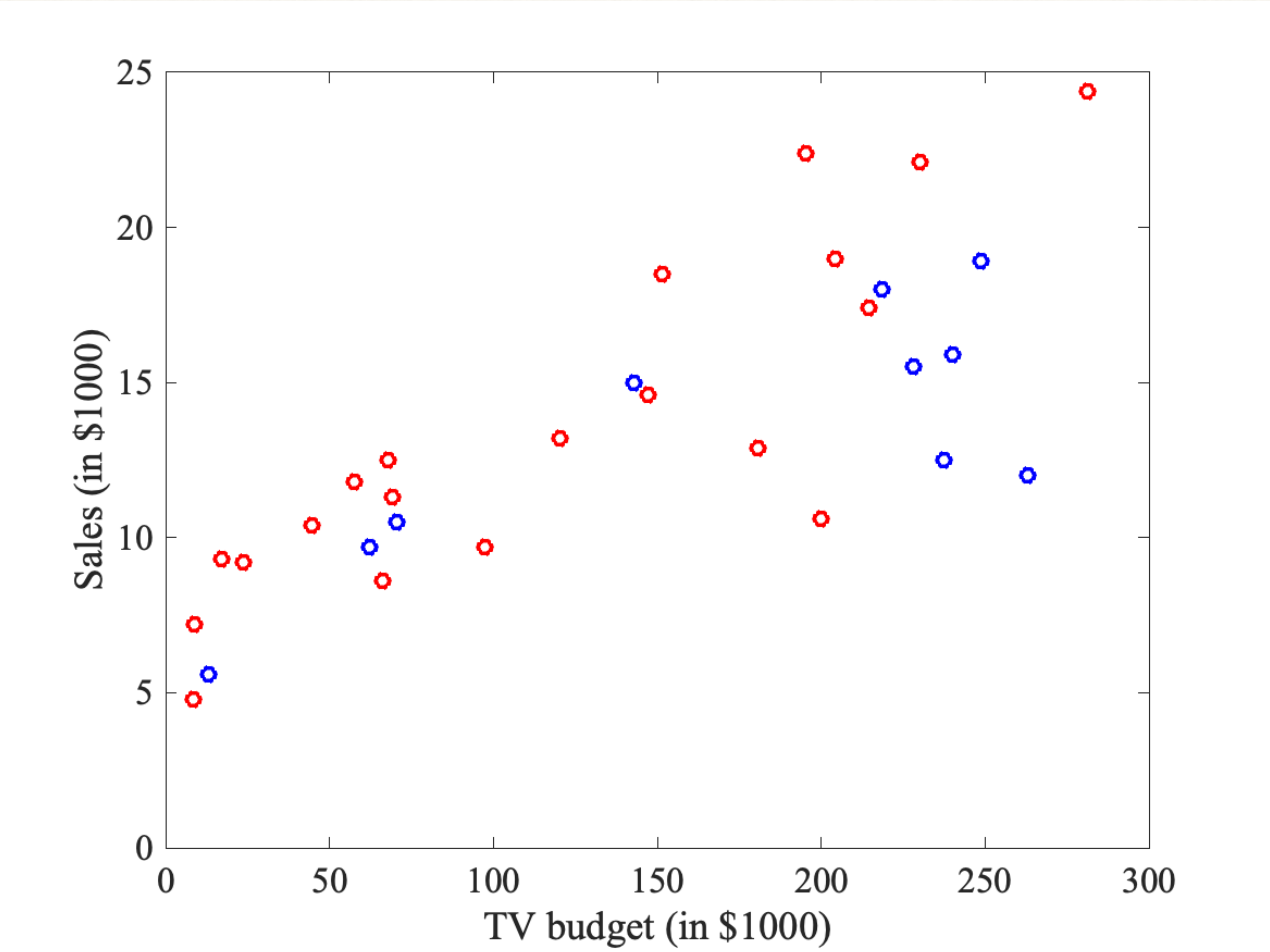
Evaluating a Model

Start with some data (x,y):



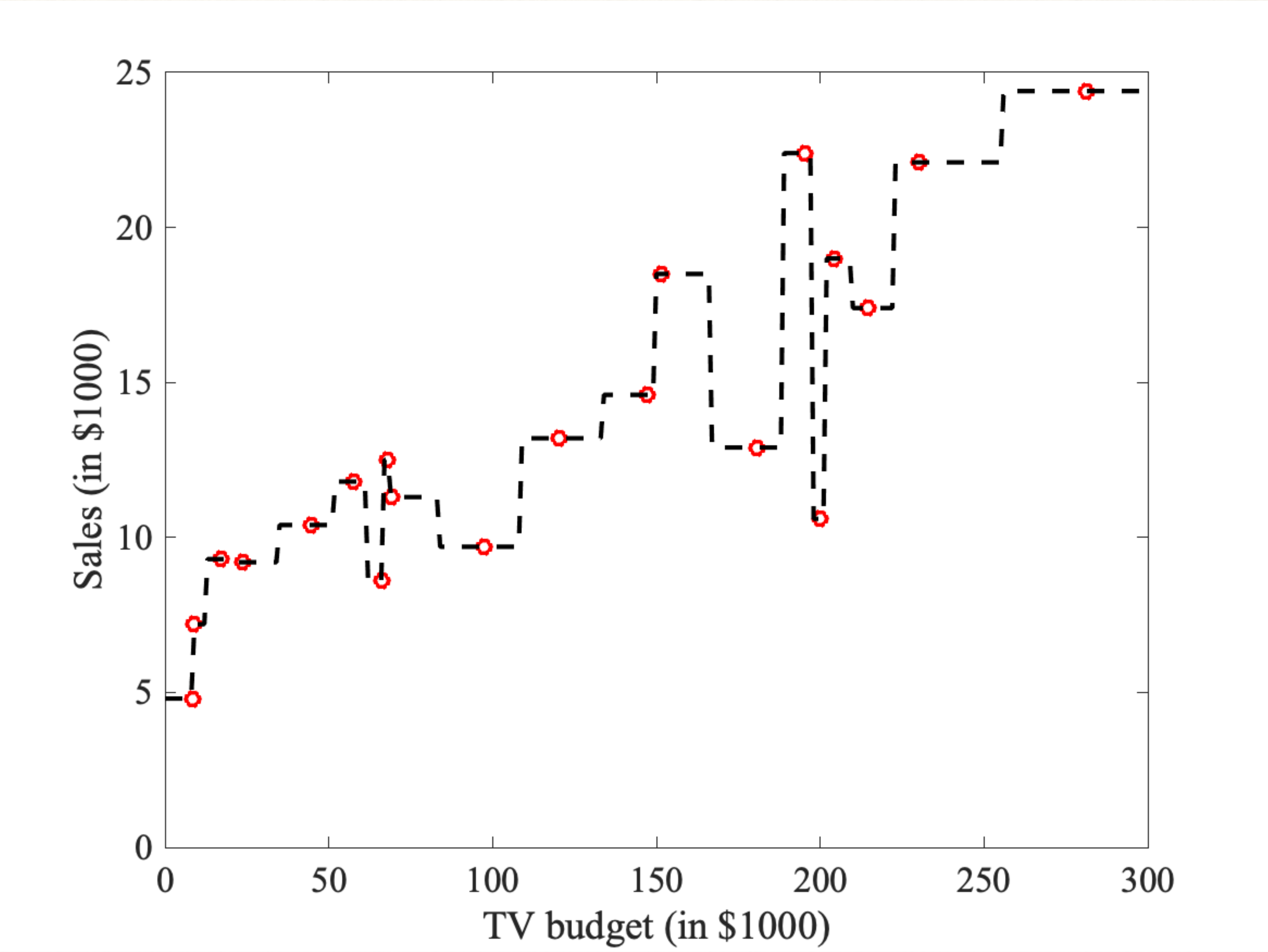
Evaluating a Model

Divide data into a training set (red) and a test set (blue):



Evaluating a Model

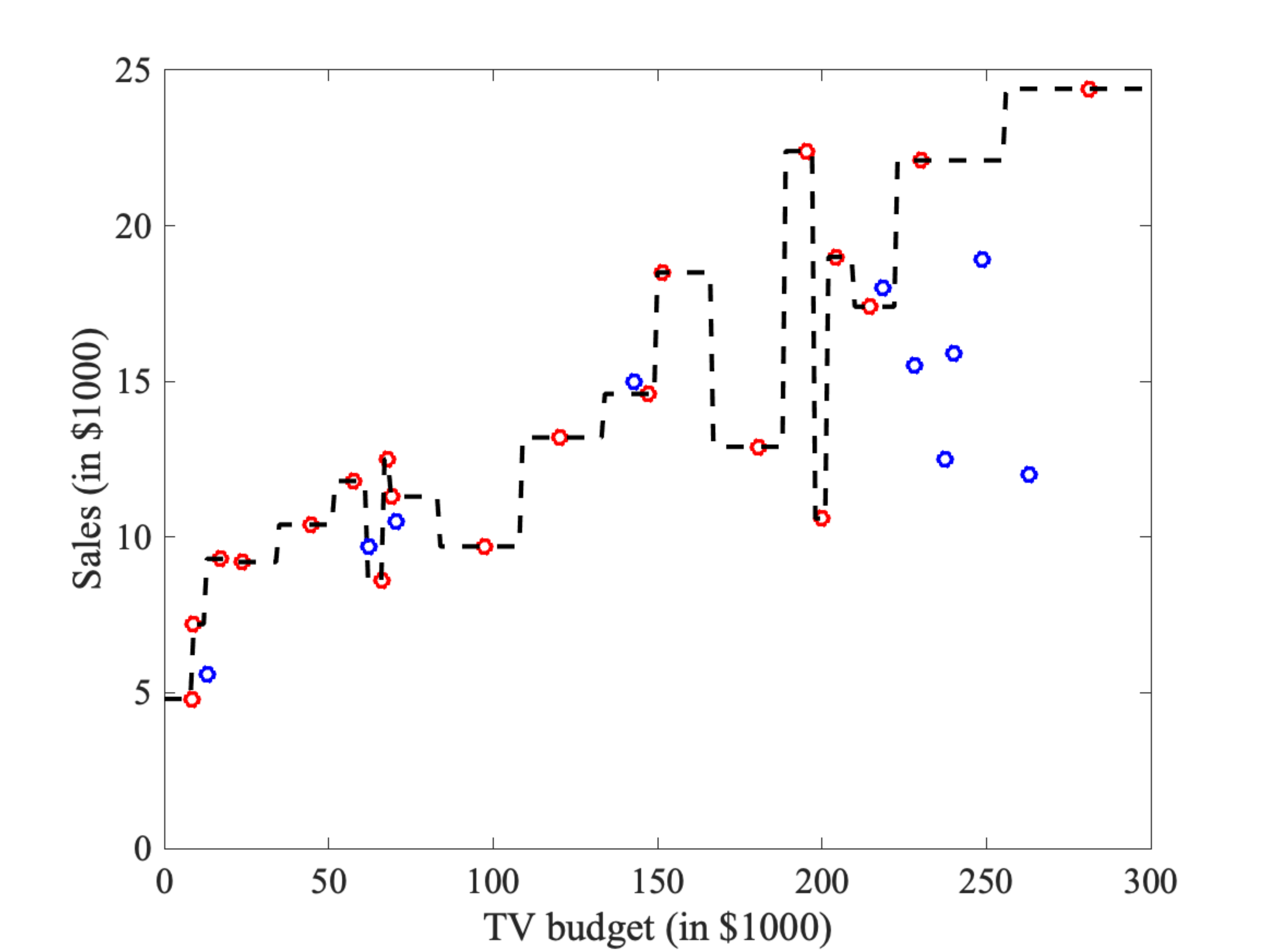
1. Build a model based on training set (here a 1-Neighbor model):



Evaluating a Model

1. Build a model based on training set (here a 1-Neighbor model)

2. Add test data

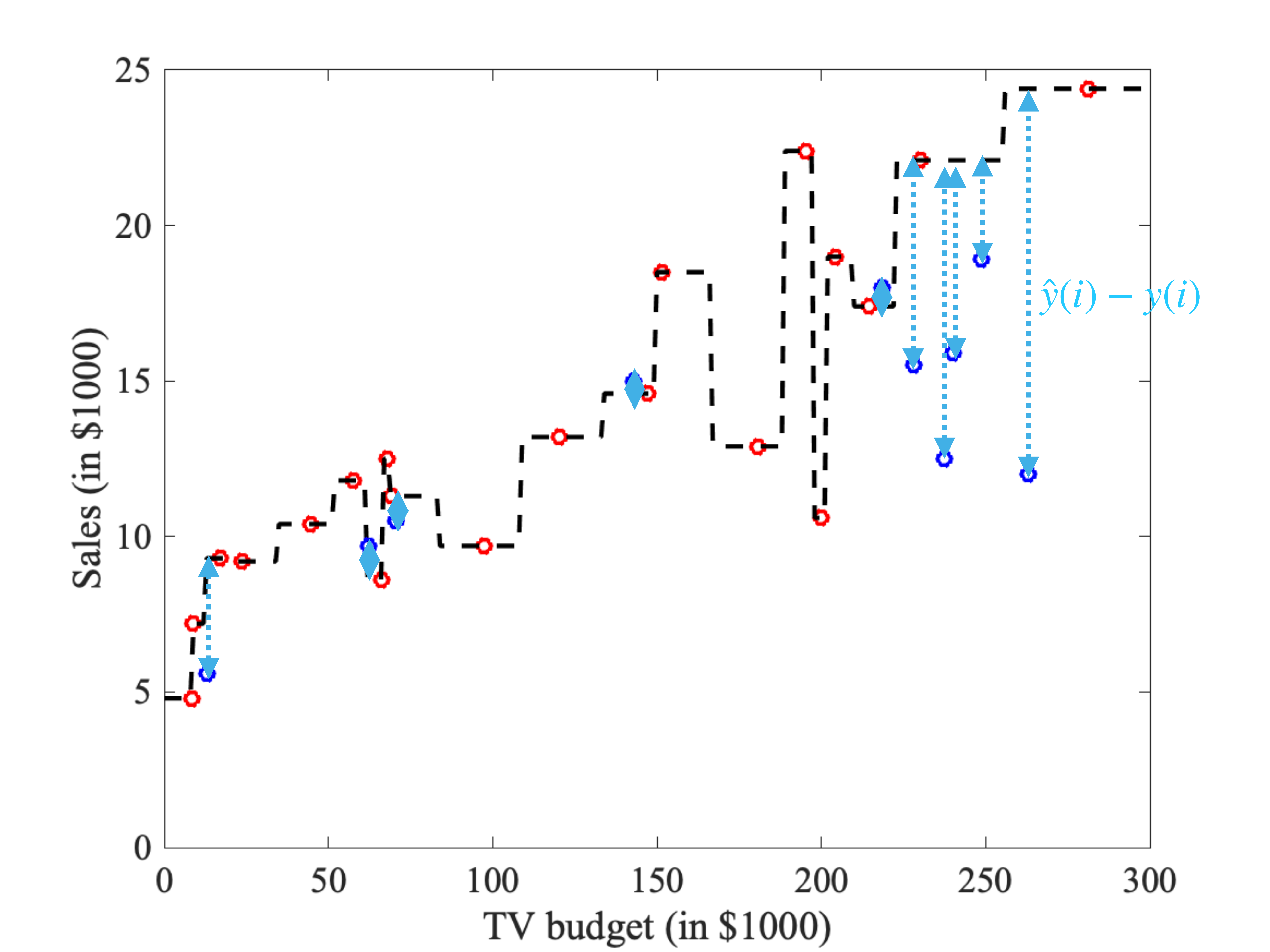


Evaluating a Model

1. Build a model based on training set (here a 1-Neighbor model)

2. Add test data

3. Compute residuals for the N test data $(\hat{y}(i) - y(i))$



Evaluating a Model

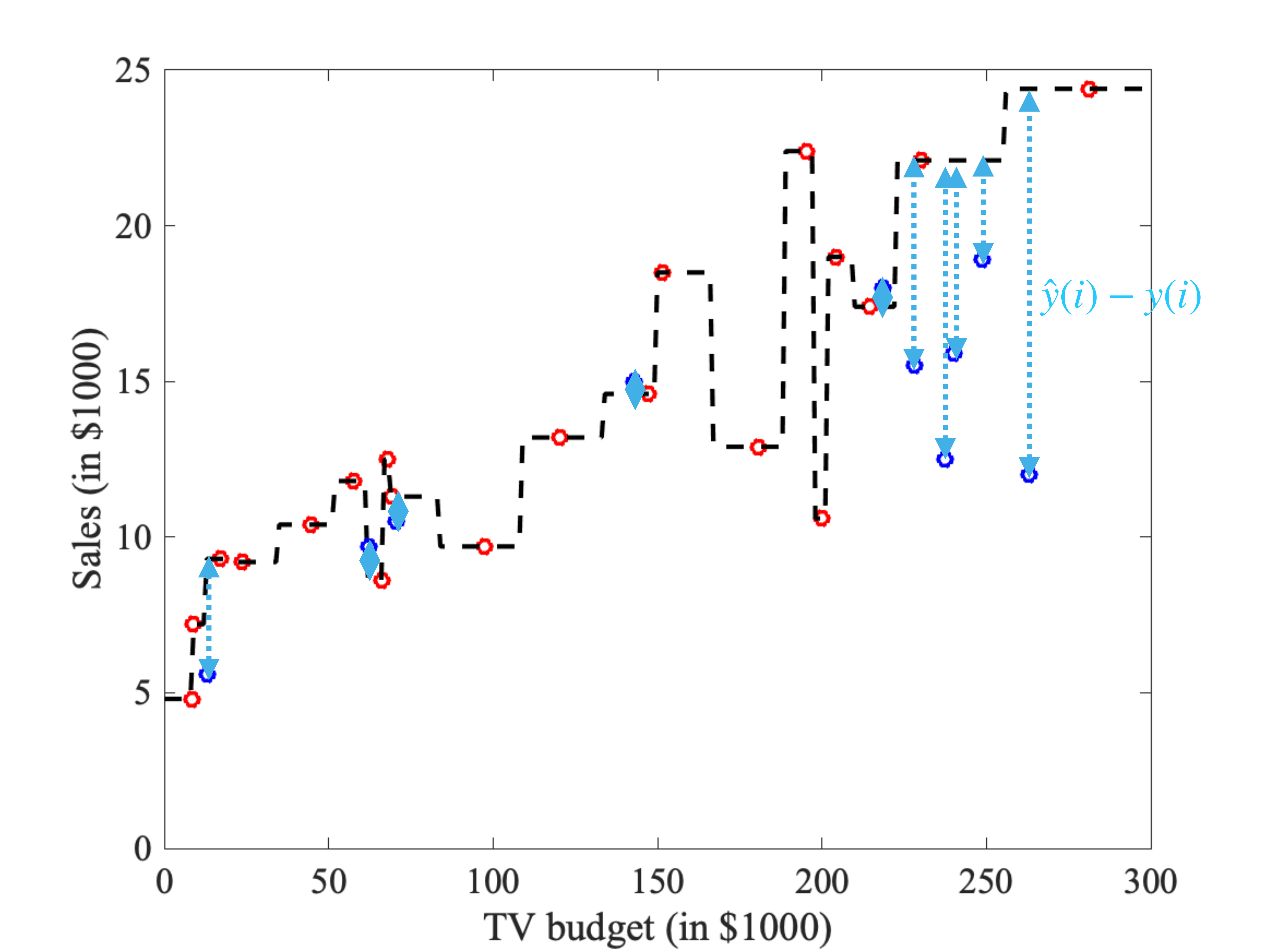
1. Build a model based on training set (here a 1-Neighbor model)

2. Add test data

3. Compute residuals for the N test data $(\hat{y}(i) - y(i))$

4. Compute the mean square error, also called loss function

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}(i) - y(i))^2$$



Evaluating a Model

Note: the mean square error is not the only possible loss function! Other possibilities:

- Mean square error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}(i) - y(i))^2$$

- Root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}(i) - y(i))^2}$$

- Maximum absolute error

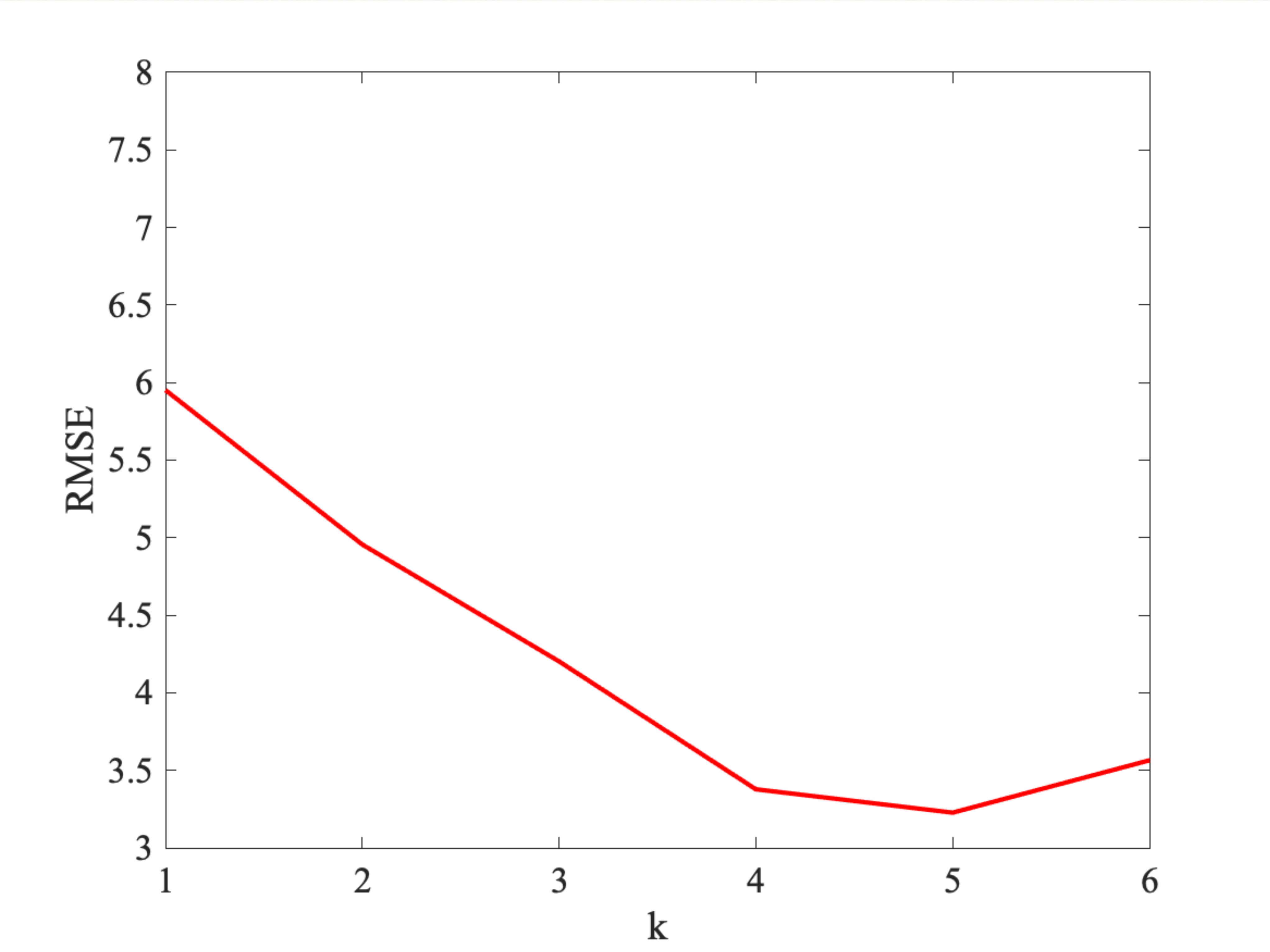
$$MAE = \max_{i\in[1,N]} |\hat{y}(i) - y(i)|$$

- Average absolute error

$$AAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}(i) - y(i)|$$

Comparing models

Compute RMSE for multiple models and plot as a function of k:

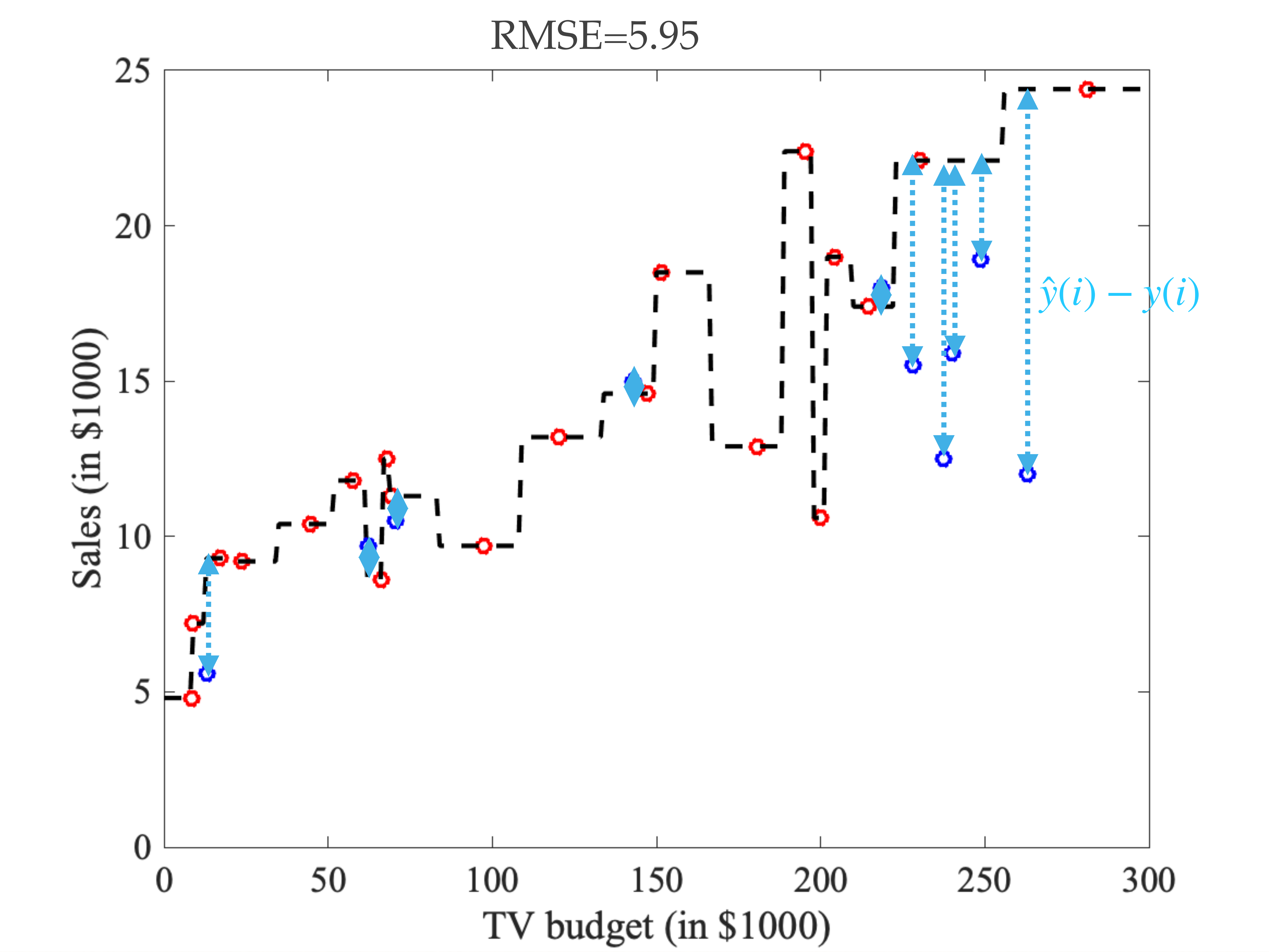


k=5 seems to be the best model

Model Fitness

Recall that for a given model, we compute the mean square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}(i) - y(i))^2}$$

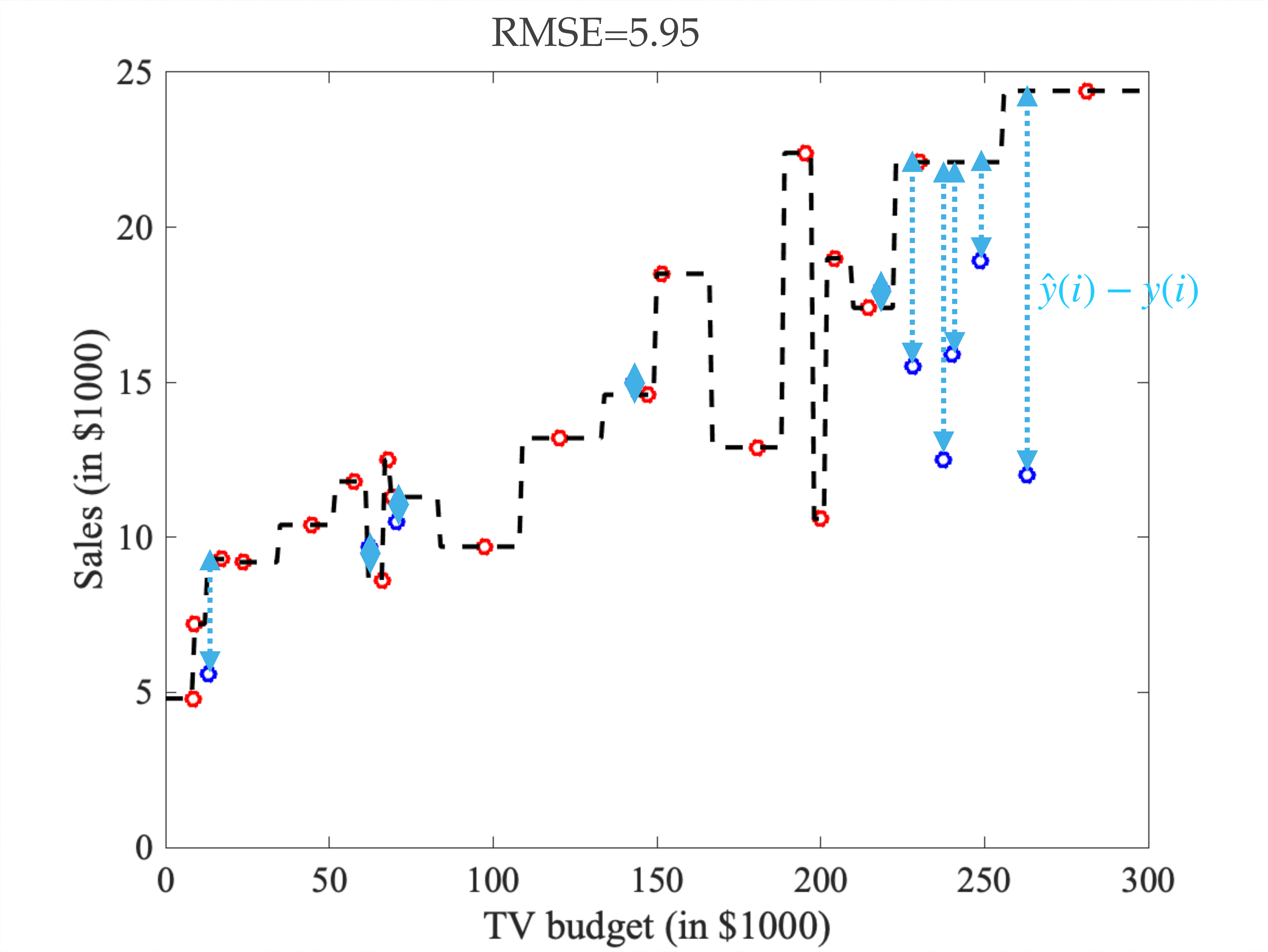


Model Fitness

Recall that for a given model, we compute the mean square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\hat{y}(i) - y(i))^2}$$

However, RMSE depends on the scale of the values y!
Here y are expressed in units of "1000 dollars";
If instead we had used dollars,
RMSE=5950



Model Fitness

To normalize the "fitness" score:

Consider the test set with values (x_i^t, y_i^t) for $i \in [1,N]$

We consider three models:

- The simplest model where each value are predicted as the average of the test set values:

$$\hat{y}^s(i) = \frac{1}{N}\sum_{i=1}^{N} y^t(i)$$

- The "best" model where each value is exact

$$\hat{y}^b(i) = y^t(i)$$

- The current model M that we want to evaluate

$$\hat{y}^M(i)$$

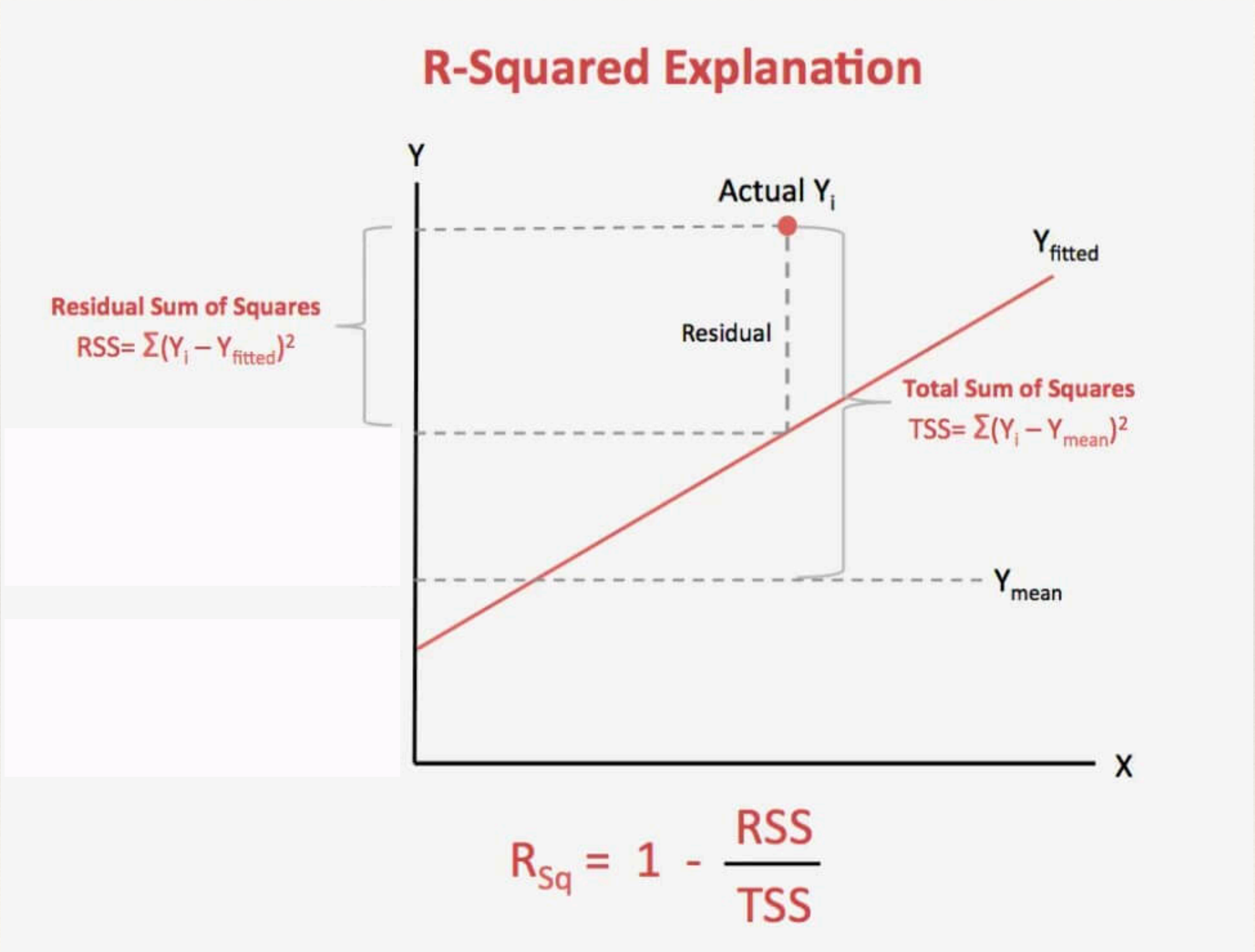
Model Fitness

To normalize the "fitness" score:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (\hat{y}^M(i) - \hat{y}^B(i))^2}{\sum_{i=1}^{N} (\hat{y}^S(i) - \hat{y}^B(i))^2}$$

- If our model is as good as the simple model, based on the average, then $R^2 = 0$
- If our model is perfect then $R^2 = 1$
- R^2 can be negative if the model is worst than the simple model (average). This can happen !

Model Fitness



RMSE or R^2?

- Both RMSE and R^2 quantify how well a model fits a dataset.
- The RMSE tells us how well a regression model can predict the value of the response variable in absolute terms while R^2 tells us how well a model can predict the value of the response variable in percentage terms.
- It is useful to calculate both the RMSE and R^2 for a given model because each metric gives us useful information.