Quantitative Social Science in the Age of Big Data and AI

Lecture 9: Introduction to Machine Learning

Zhaopeng Qu
Hopkins-Nanjing Center
May 19 2025

Outline

Machine Learning and Prediction

Introduction

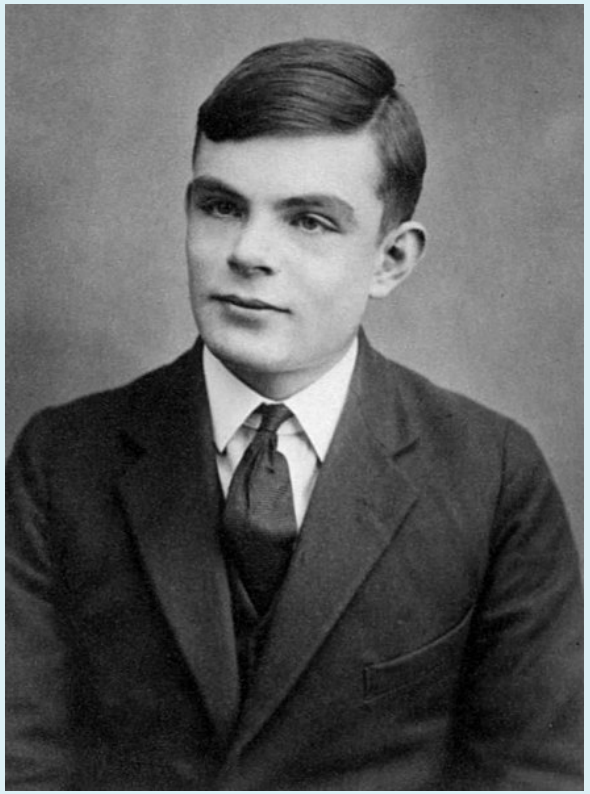
- Machine learning is originally a branch of computer science and statistics.

> "...[M]achine learning is a field that develops algorithms designed to be applied to datasets, with the main areas of focus being prediction (regression), classification, and clustering or grouping tasks." by Susan Athey(2018)

- The *learning* part comes from the fact that we do not specify how exactly the computer should predict y from x.

- In general, this means that we abstract from the underlying models (biologic, economic, etc.) that creates the outcome that we want to predict.

The Origins of AI and Machine Learning



- Alan Truing(1912-1954),English mathematician and logician,widely considered to be the father of theoretical computer science and artificial intelligence.

- Authur Samuel(1901-1990),American computer scientist,pioneerly popularized the term machine learning in 1959.

Terminology: Econometrics V.S ML

	Causal inference	Machine learning
Topic	a causal effect	a learning problem
Object	estimate β	Fitted value of $\hat{y}$
How	run an estimation	train an algorithm
Criterion	Unbiasedness and Consistency	Optimal fit
Evaluation	Conceptual Key assumptions	Cross-validate fit
Question	causal or not	accurate or not
Variables	an independent or treatment Variable	a feature
Variables	a continuous dependent variable	a response
Variables	a categorical dependent variable	a label

- Many similarities, but also some differences.

Machine Learning: Algorithms

- Any algorithm that maps *features*(independent variables) into a *prediction*(dependent variable) can be thought of as within the realm of machine learning.
- There are many machine learning algorithm. The best methods vary with the particular data application.
  - Regression: OLS,LASSO,Ridge
  - Classification: logit,probit
  - Decision trees and random forests
  - Neural networks and support vector machines
  - ...
- In many cases, the theoretical properties (e.g. convergence and limit distribution) of these algorithms are even unknown but *that is not the point*.

Machine Learning: A broad classification

1. Supervised learning: We have data on both an *outcome* Y and *explanatory variables* X.

- The goal is to predict Y from X, and many methods can be used to do this.
    - Regression: if Y is continuous
    - Classification: if Y is discrete
    - K-Nearest Neighbors
    - Decision Trees
    - Random Forests
    - Support Vector Machines
    - Neural Networks
- Applications:
    - Predicting electricity demand from temperature
    - Predicting presidential election from economic indicators and news coverage
    - Predicting spam emails from email content

Machine Learning: A broad classification

1. Unsupervised learning: we have no data on Y, only on X.
    - Cluster Analysis
    - Principle Component Analysis(PCA)
    - Latent Dirichlet Allocation(LDA)
- Applications:
    - Image recognition
    - Text classification
    - Clustering customers

1. Advanced Methods:

- Reinforcement learning
- Deep learning
- Applications:
    - Game playing
    - Autonomous driving
    - LLMs like ChatGPT

Our focuses

Main Content

- Basic ideas of ML
- ML algorithms for prediction
- How and when to apply ML methods in QSS research.

Not about

- Cutting-edge ML techniques
- Computational aspects
- Distributed computation systems for large data

Supervised learning

Introduction: Supervised Learning

Suppose the the relationship between x and y can be written as an additive error model:

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

- where $f()$ is some fixed but unknown function of X, which it represents the systematic relationship between X and Y.
- And ϵ represents *idiosyncratic deviations* from this systematic relationship, so it satisfies

$$E(\epsilon \mid X) = 0 \text{ and } E(\epsilon) = 0$$

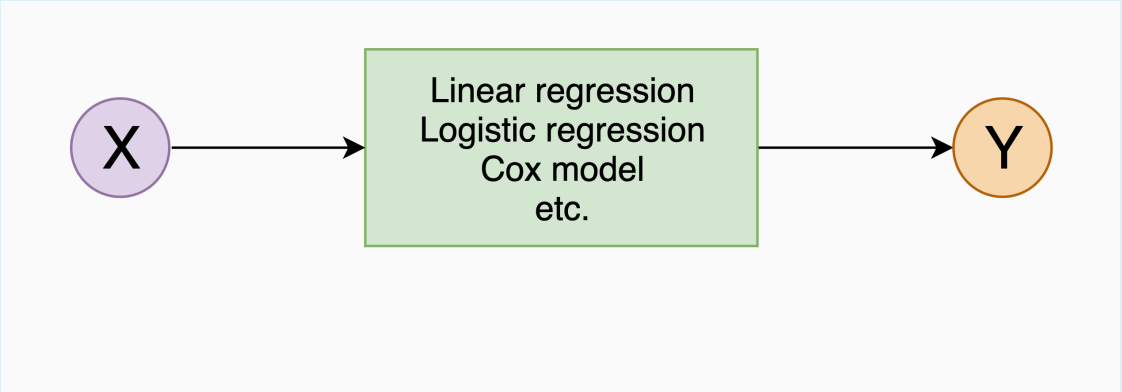
Causal inference v.s Prediction

1. Causal inference: How do changes in X affect Y? $\beta = \frac{\partial Y}{\partial X}$
2. Prediction: Predict Y using our estimated $f(X)$, i.e.,

$$\hat{Y} = f(\hat{X})$$

The Objective of Supervised Learning

- In supervised learning, we want to make a prediction about the response Y based on features X.
- Because it helps us to make a prediction, it is useful to estimate $f(\cdot)$, which represents the systematic relationship between features(X) and the response(Y).



- However, for prediction we do not care about $f(\cdot)$ itself. We can treat it as a *black box*, and any approximation $\hat{f}(\cdot)$ that yields a good prediction is *good enough*.
  - Whatever works, works

Example: predicting electricity demand

- ERCOT (Electric Reliability Council of Texas) operates the electricity grid for 75% of Texas by area.

Example: predicting electricity demand

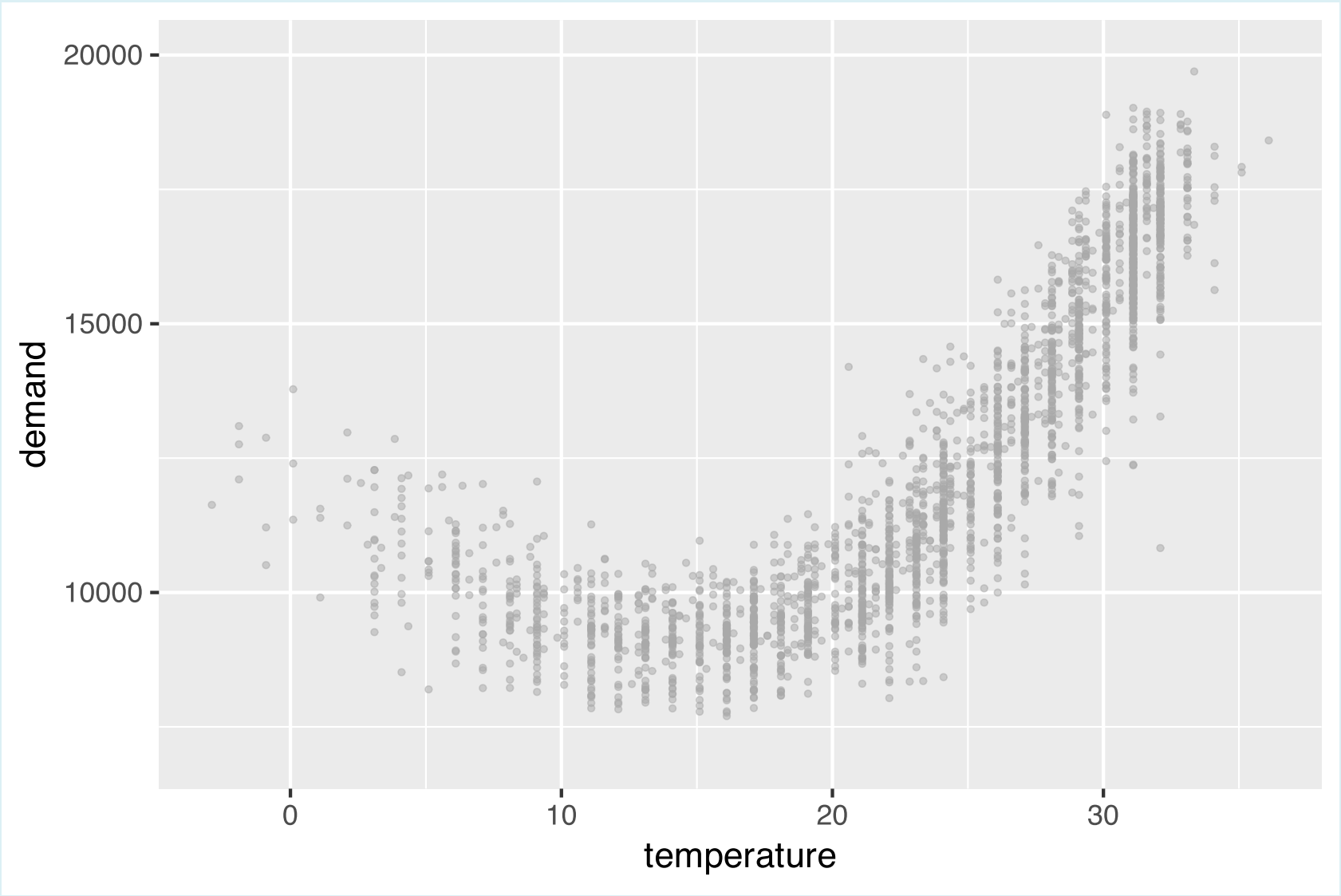
The 8 ERCOT regions are shown at left. We'll focus on a basic prediction task:

- Y = demand (megawatts) in the Coast region at 3 PM, every day from 2010-2016.
- x = average daily temperature at Houston's Hobby Airport (Celsius degrees)

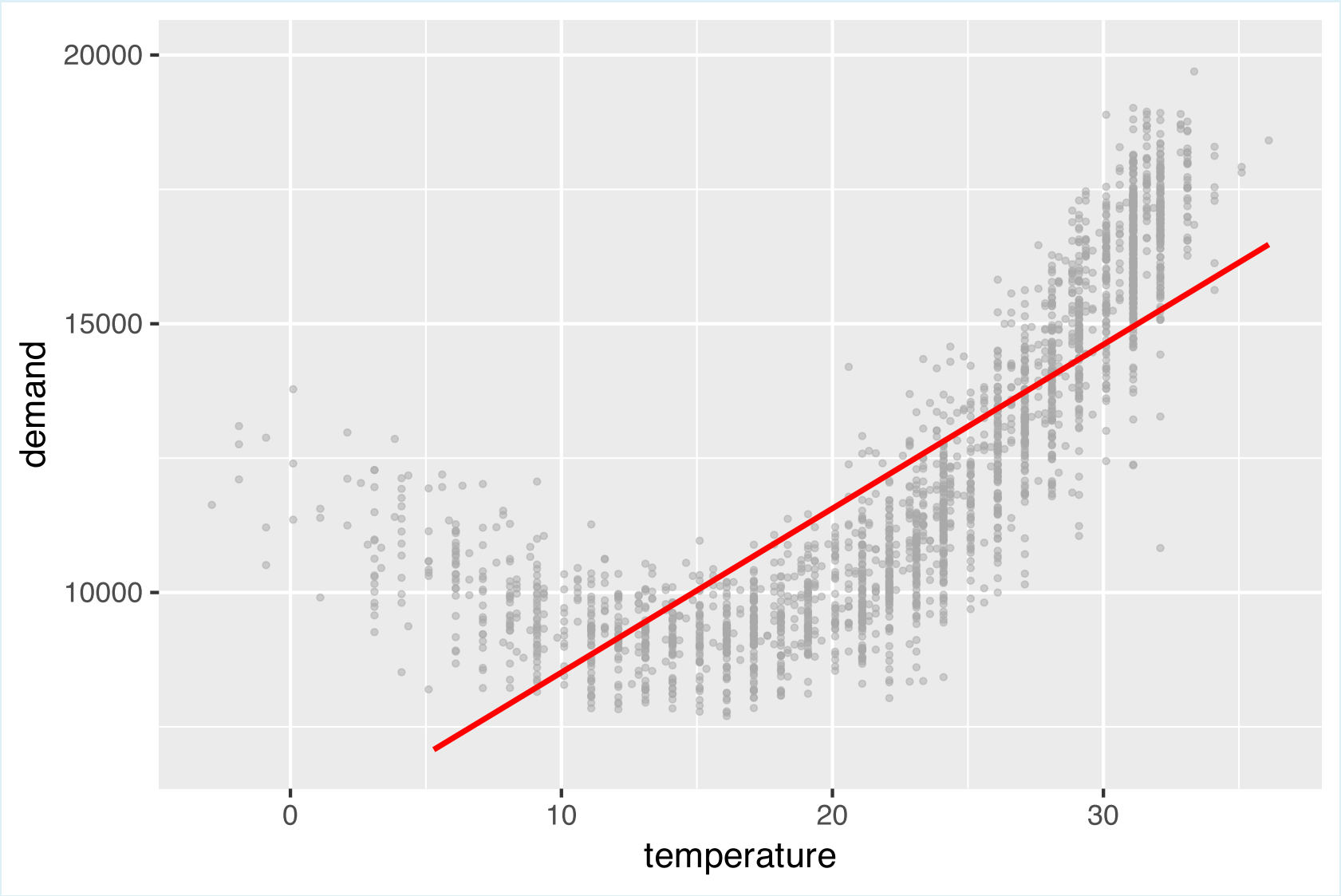
--

Time	KHOU	COAST
1/1/10 15:00	7.1	8222.029
1/2/10 15:00	9.1	8379.872
1/3/10 15:00	6.1	8679.087
1/4/10 15:00	4.1	10273.567

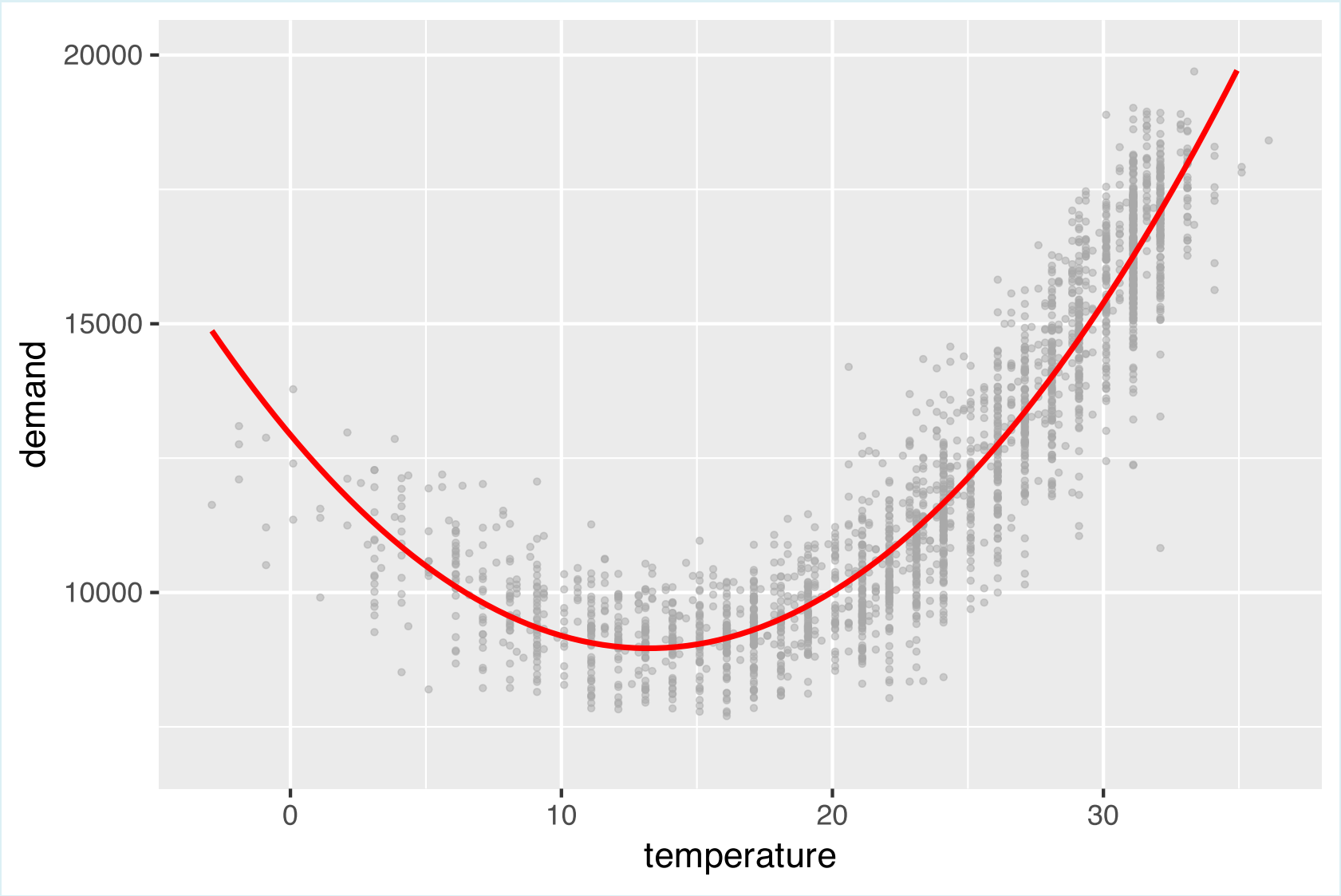
Demand v.s Temperature



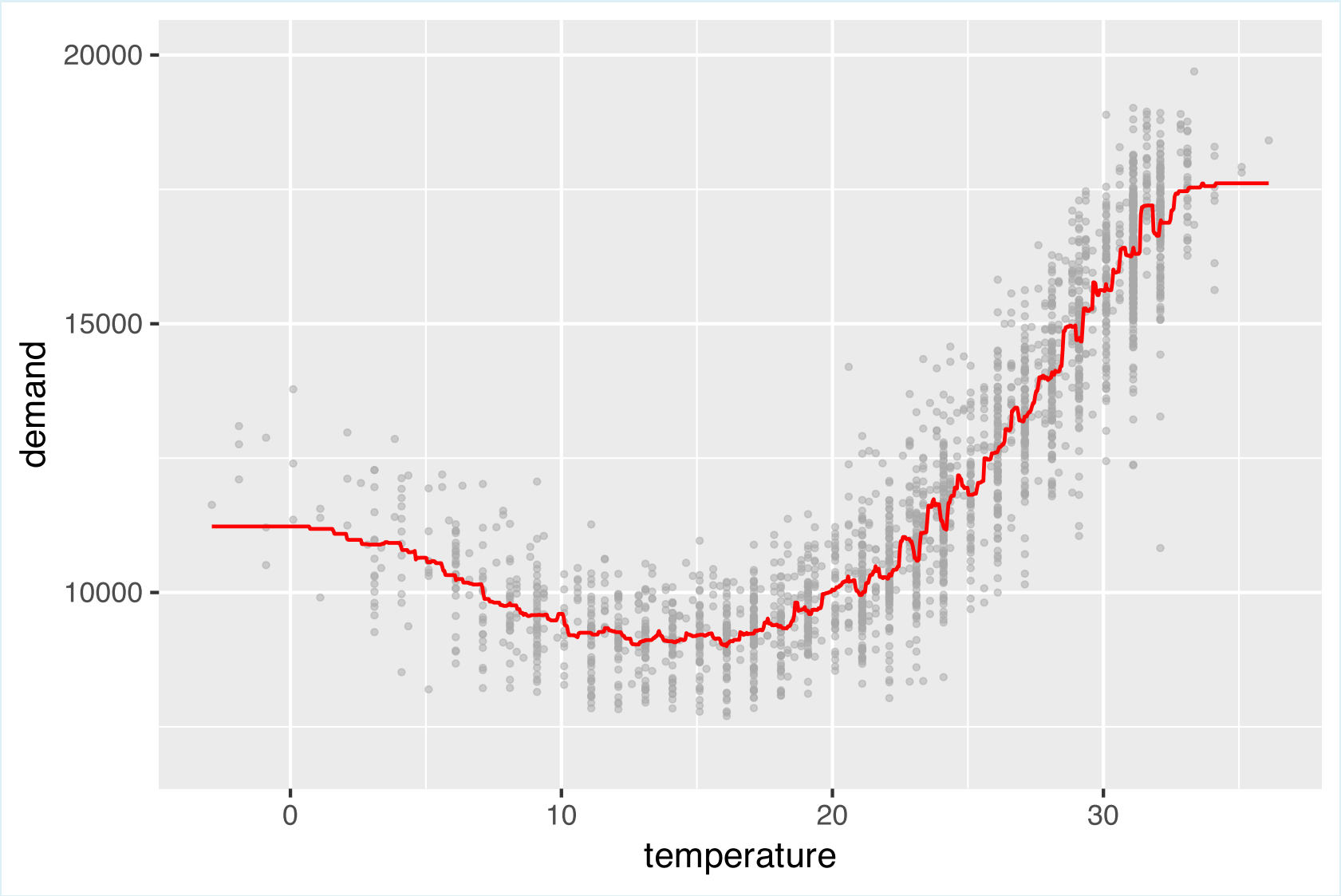
A linear model?



A quadratic model?



How about this model?



Formally: Optimal Objective

- Question: *How to choose the best prediction model?*
- Formally, a supervised learning algorithm takes as an input a loss function and searches for *a function within a function class* that has a low expected prediction loss on a new data point from the same distribution.
- A very common loss function in a regression setting is the mean squared error (MSE), thus

$$MSE = \frac{1}{N}\Sigma_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

  - *Do you feel familiar with this loss function?*
  - It is the same as the mistaken-function in OLS regression we have learned.
- The optimal prediction is the one that minimizes the MSE just like the OLS regression.

Error Decomposition

- The MSE is a sample concept. The population analogue is called the expected mean-squared error(EMSE), thus expectation of MSE over the population.

- Because $E(\epsilon \mid x) = 0$ and $E(\epsilon) = 0$, then

$$
\begin{aligned}
EMSE &= E[Y - \hat{Y}]^2 \\
&= E[f(X) + \epsilon - \hat{f}(X)]^2 \\
&= E[(f(X) - \hat{f}(X))^2] + E[\epsilon^2] - E[2(f(X) - \hat{f}(X))\epsilon] \\
&= \underbrace{E[f(X) - \hat{f}(X)]^2}_{\textbf{Reducible error}} + \underbrace{Var(\epsilon)}_{\textbf{Irreducible error}}
\end{aligned}
$$

- We could prove that

$$
E(Y \mid X) = \arg\min_{f(X)} EMSE
$$

  (Ref: MHE-Theorem 3.1.2,pp33)

  - Thus the Conditional Expectation Function(CEF) is the best predictor of Y given X.

Unknown function form of f(X)

How to obtain the forms of CEF or $f(X)$

- Parametric: assume a particular, restricted functional form (e.g. linear, quadratic, logs, exp)

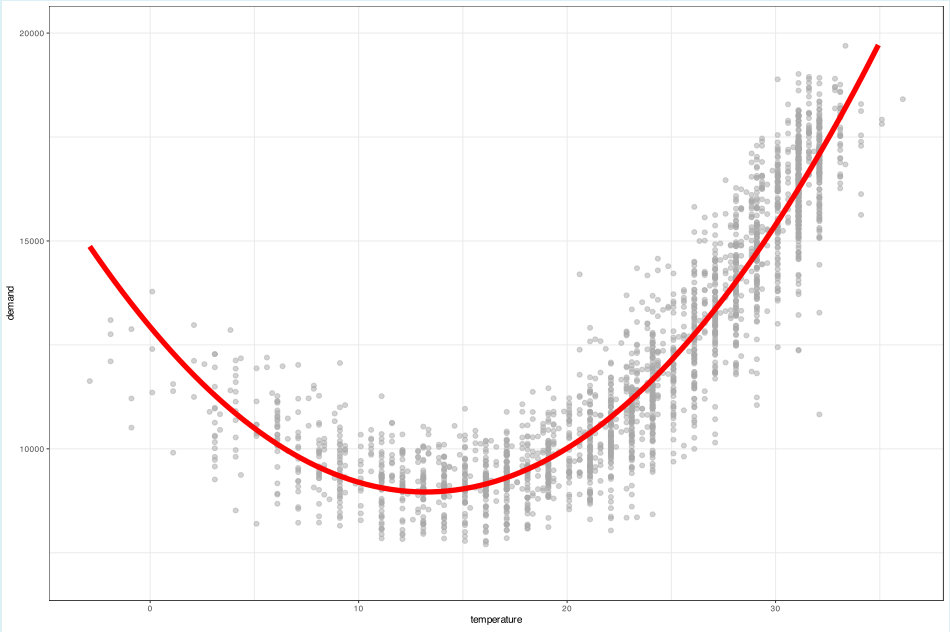
$$f(X) = g(\beta X)$$

  - The simplest one is OLS regression $f(X) = X'\beta$

- Nonparametric: flexible forms not easily described by simple math functions.

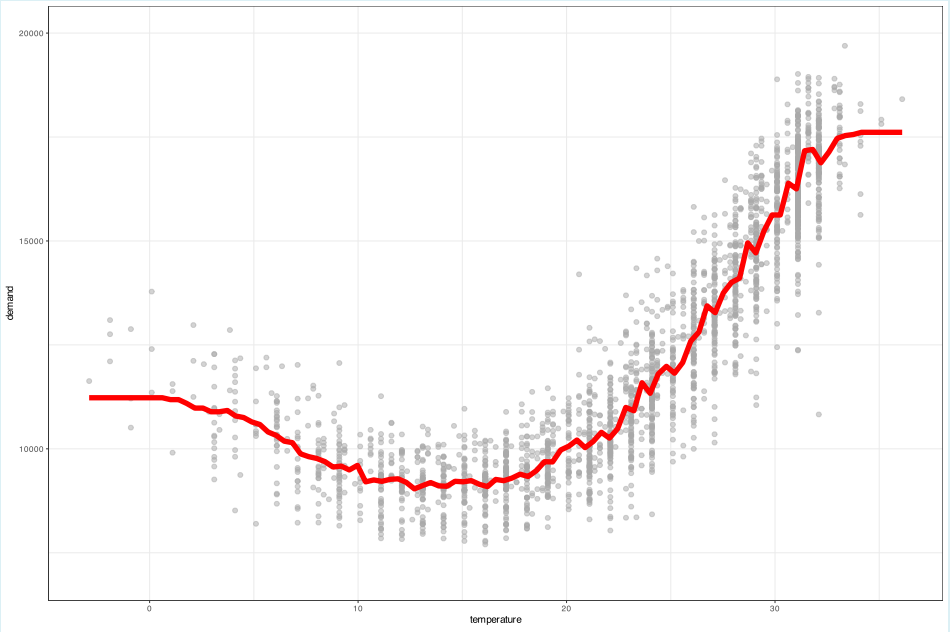
  - Matching(Nearest Neighbors)

Parametric v.s Nonparametric



Parametric: polynomial model

- $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2$



Nonparametric: k-nearest neighbors(KNN)

- $f(X) =$ average Y value of the 50 points closest to X

Estimating a parametric model: three steps

Suppose we have data in the form of (x_i, y_i) pairs. Now we want to predict y at some new point $y^\star$.

1. Choose a functional form of the model, e.g.

$$f(X) = \beta_0 + \beta_1 X$$

2. Choose a loss function that measures the difference between the model predictions $f(X)$ and the actual outcomes y. E.g. least squares:

$$L(\beta_0, \beta_1) = \sum_{i=1}^{N} (y_i - f(X_i))^2 = \sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_i))^2$$

3. Find the parameters that minimize the loss function.

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\min_{\beta_0, \beta_1} L(\beta_0, \beta_1)$$

Estimating k-nearest neighbors(KNN)

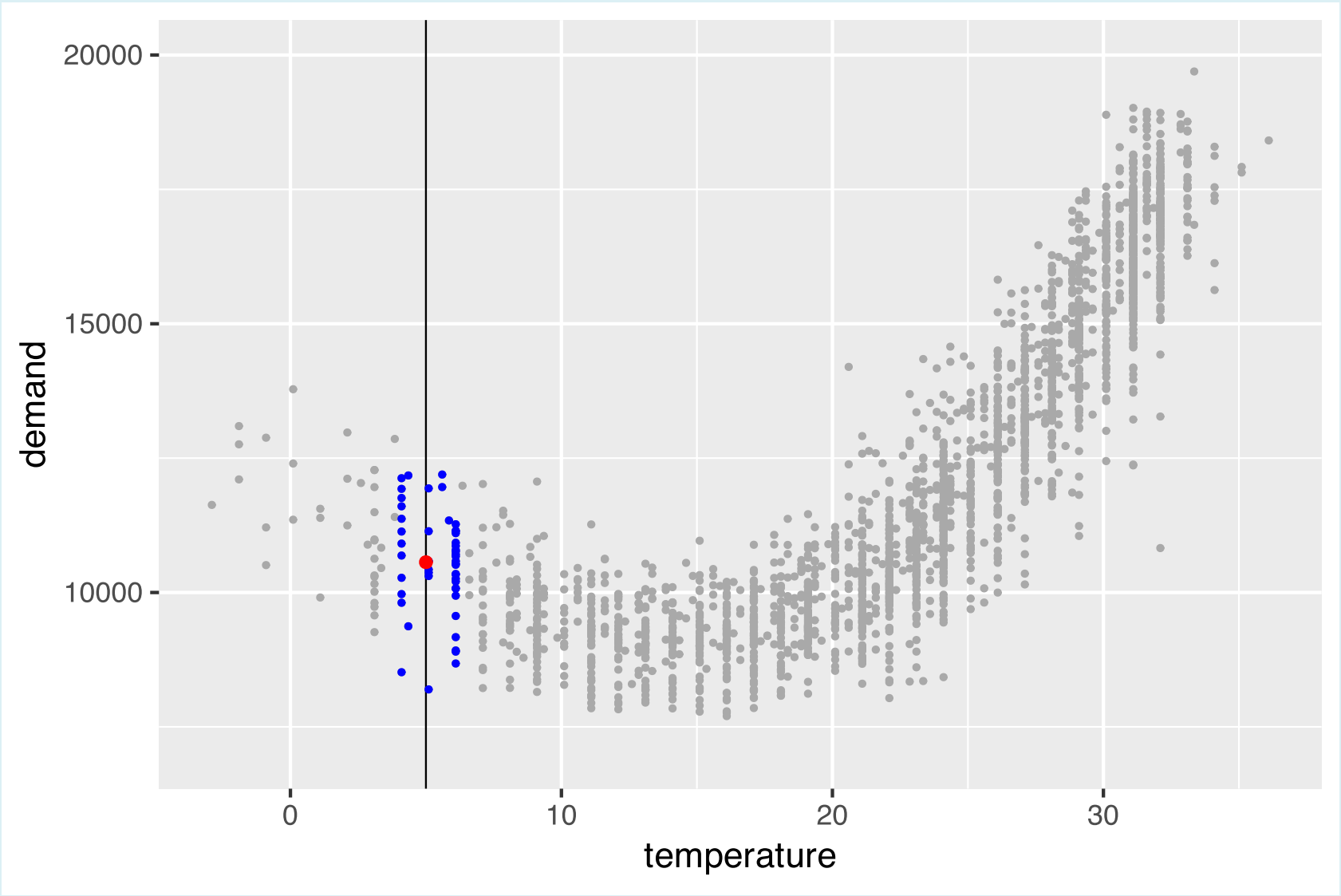
1. Pick the K points in the data whose x_i values are closest to $x^\star$. Call this neighborhood $\mathcal{N}_K(x^\star)$.

2. Average the y_i values for those points and use this average to estimate $f(x^\star)$:

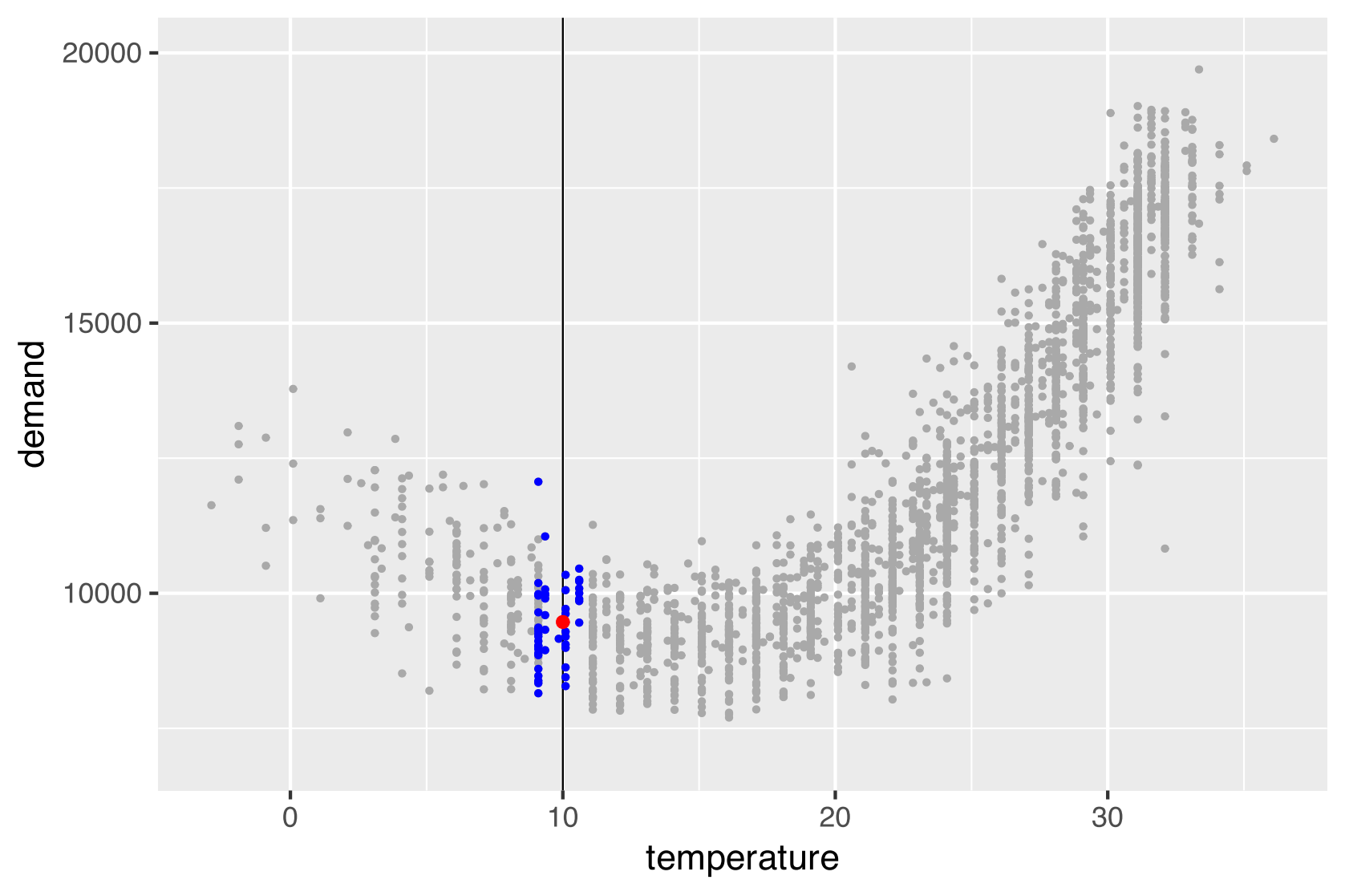
$$\hat{f}(x^\star) = \frac{1}{K} \sum_{i:x_i \in \mathcal{N}_K(x^\star)} y_i$$

- There are no explicit parameters (i.e. β's) to estimate.
- Rather, the estimate for $f(x)$ is defined by a particular *algorithm* applied to the data set.
- Suppose $K = 50$, thus we use the average of the 50 observations closest to $x^\star$ to estimate $f(x^\star)$.

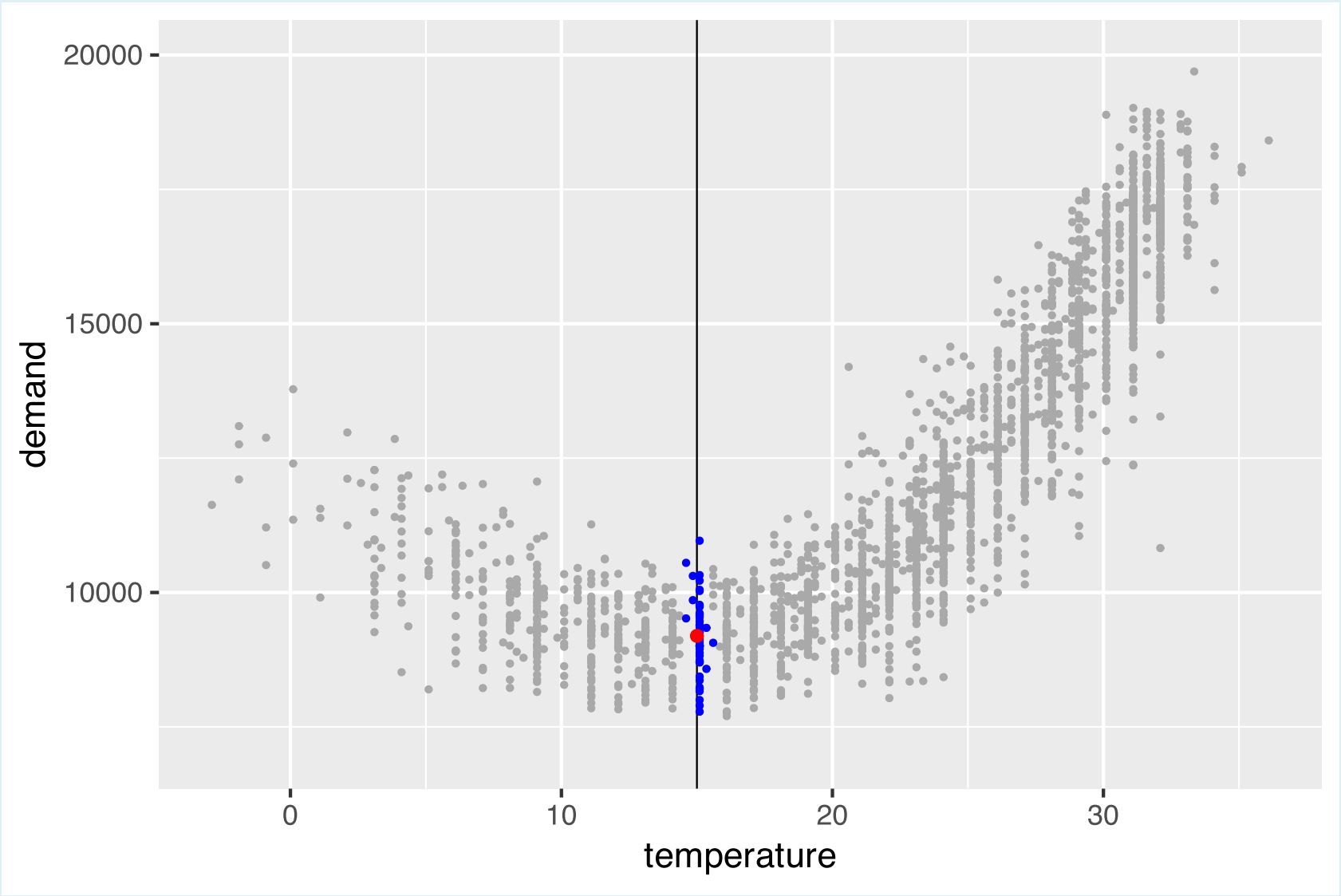
At x=5 and K=50



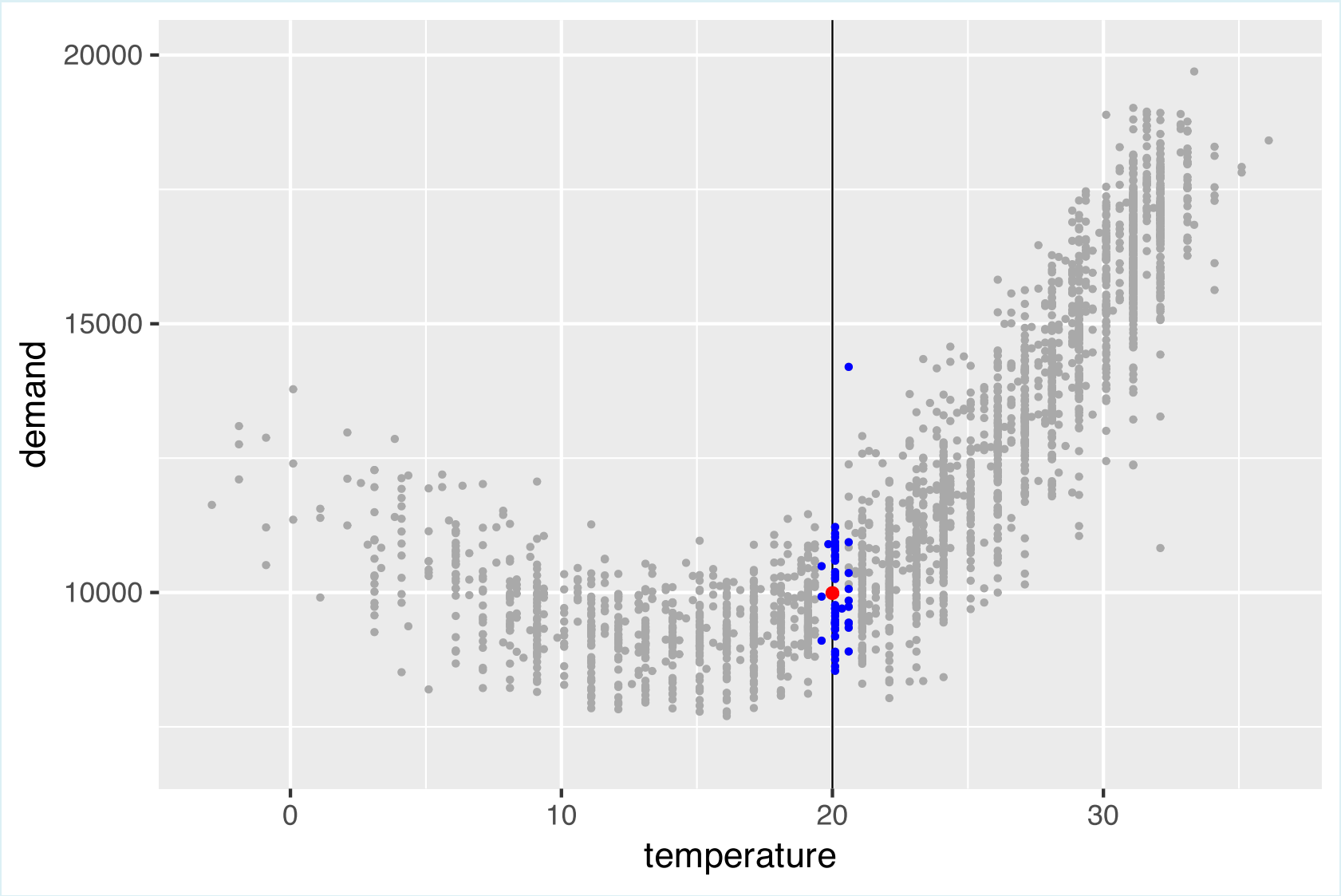
At x=10 and K=50



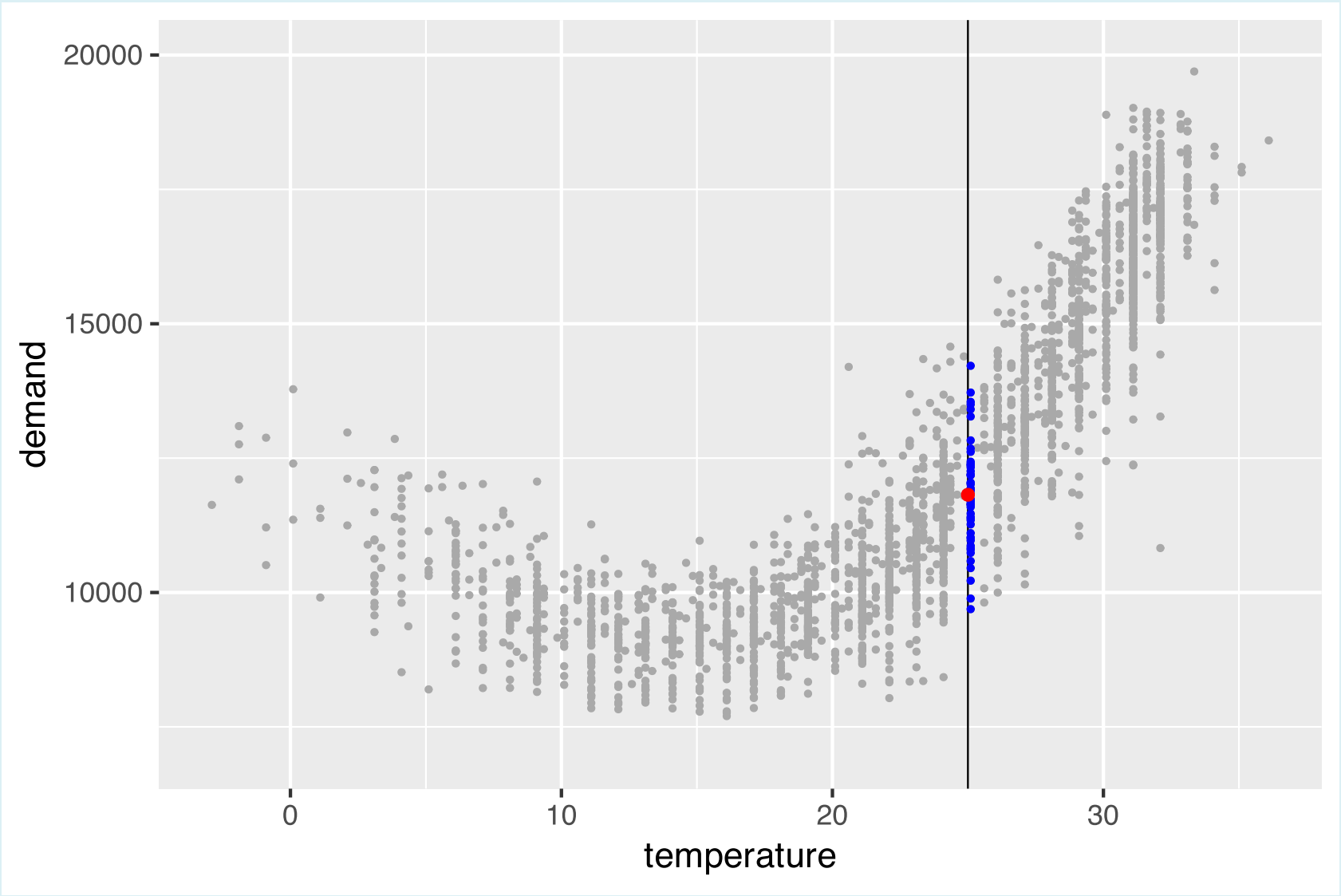
At x=15 and K=50



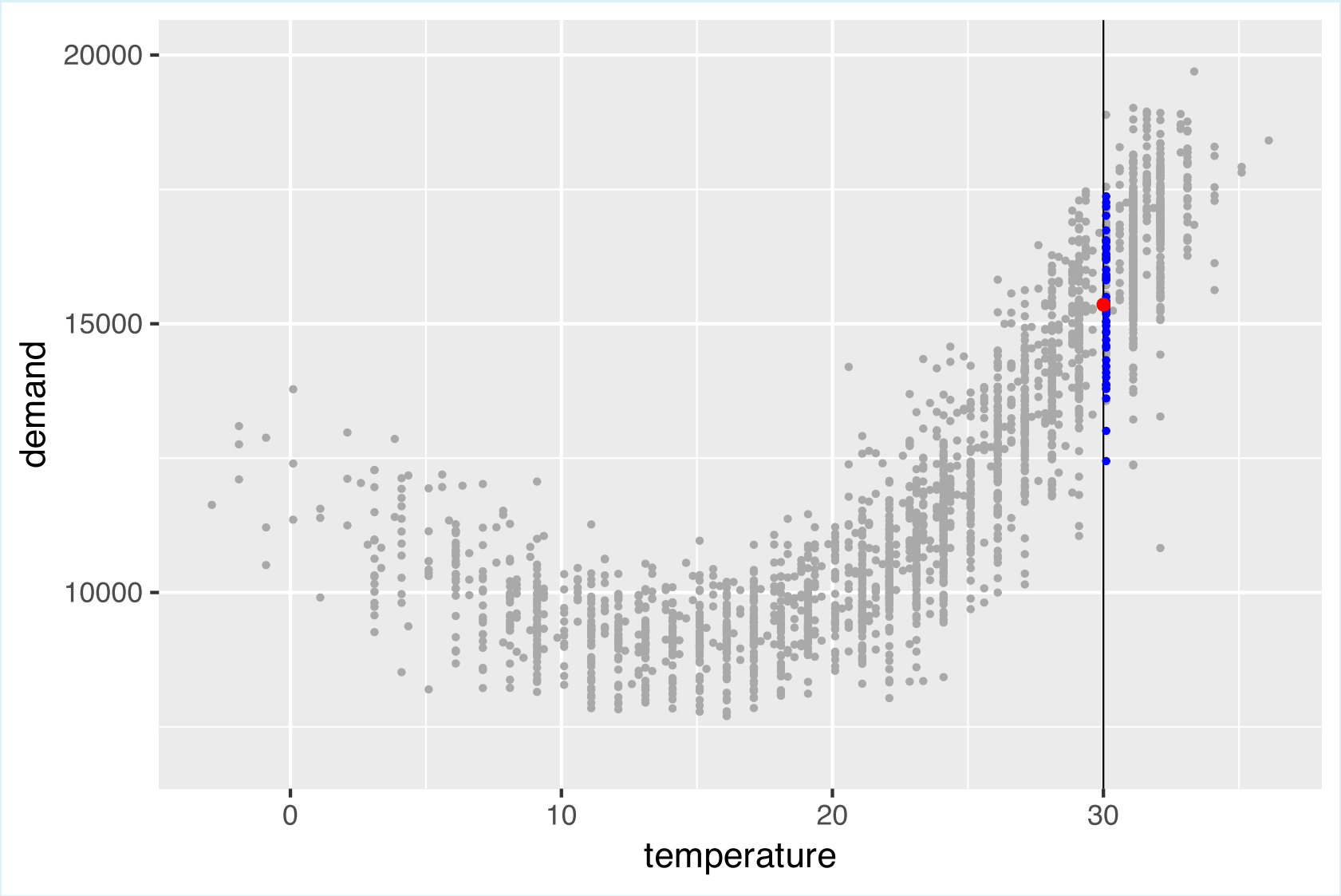
At x=20 and K=50



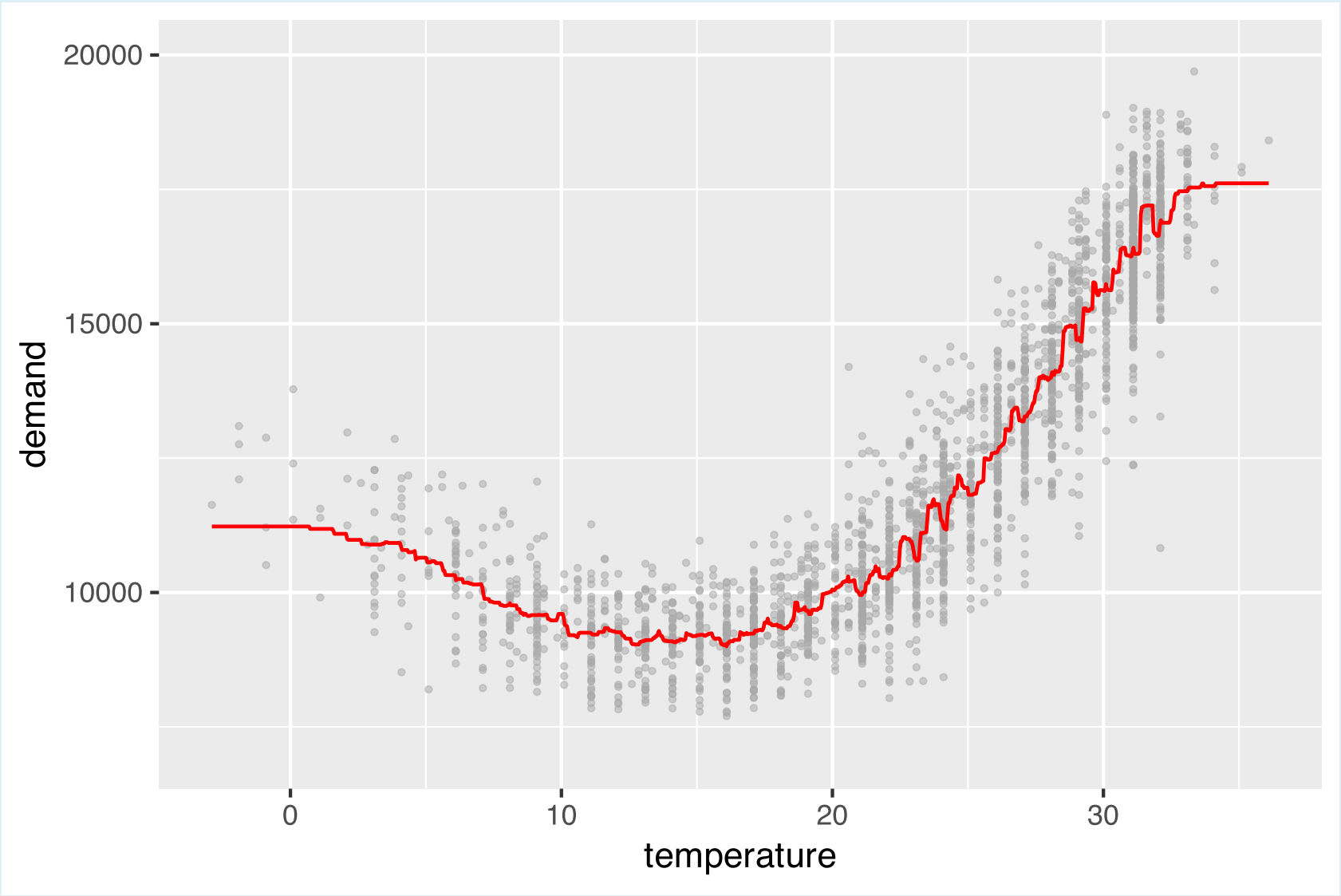
At x=25 and K=50



At x=30 and K=50



The predictions across all x values



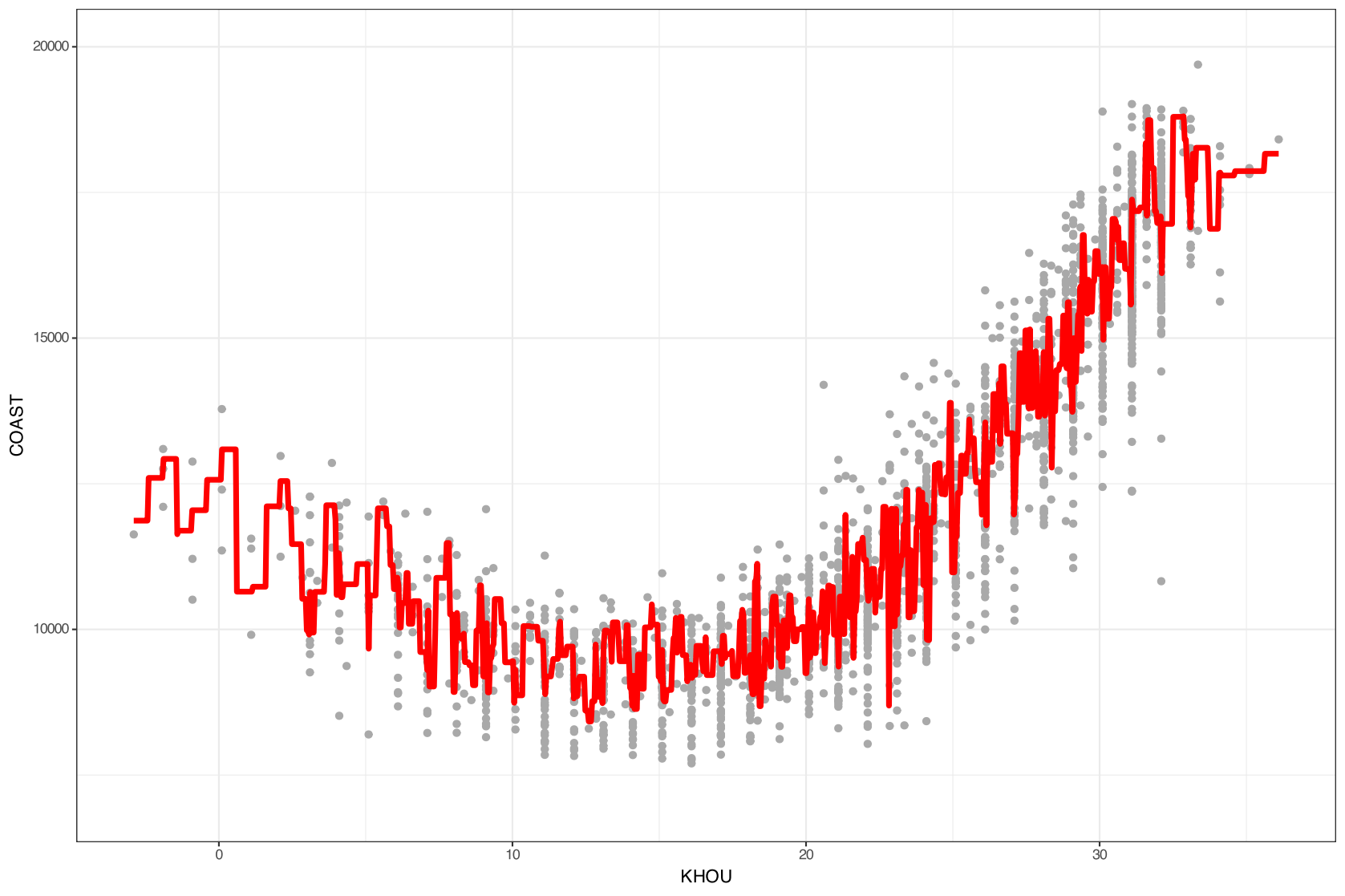
Two questions

This procedure raises two obvious questions:

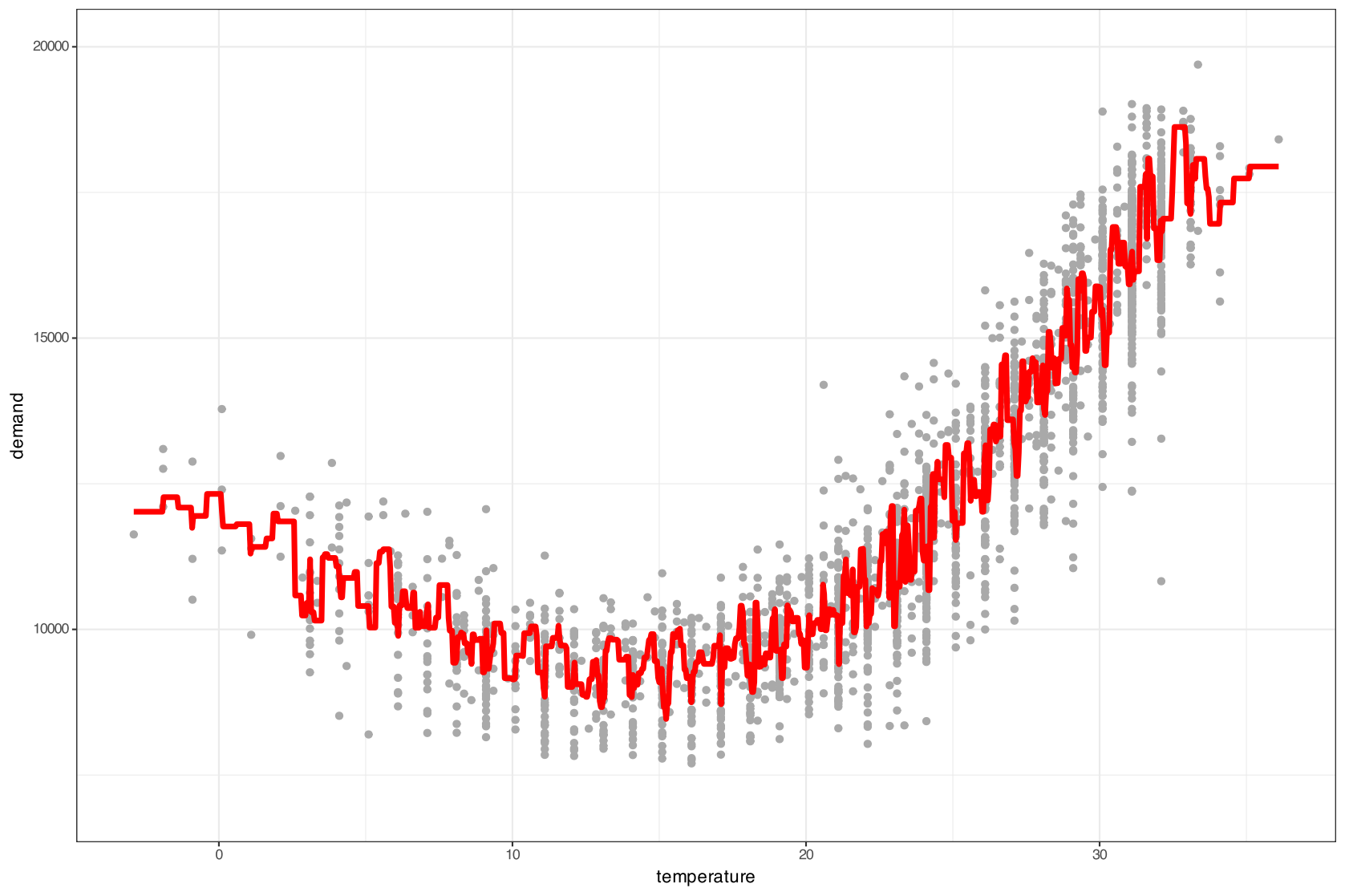
1. So why average the nearest $K = 50$ neighbors? Why not $K = 2$, or $K = 200$?

2. And if we're free to pick any value of K we like, how should we choose?

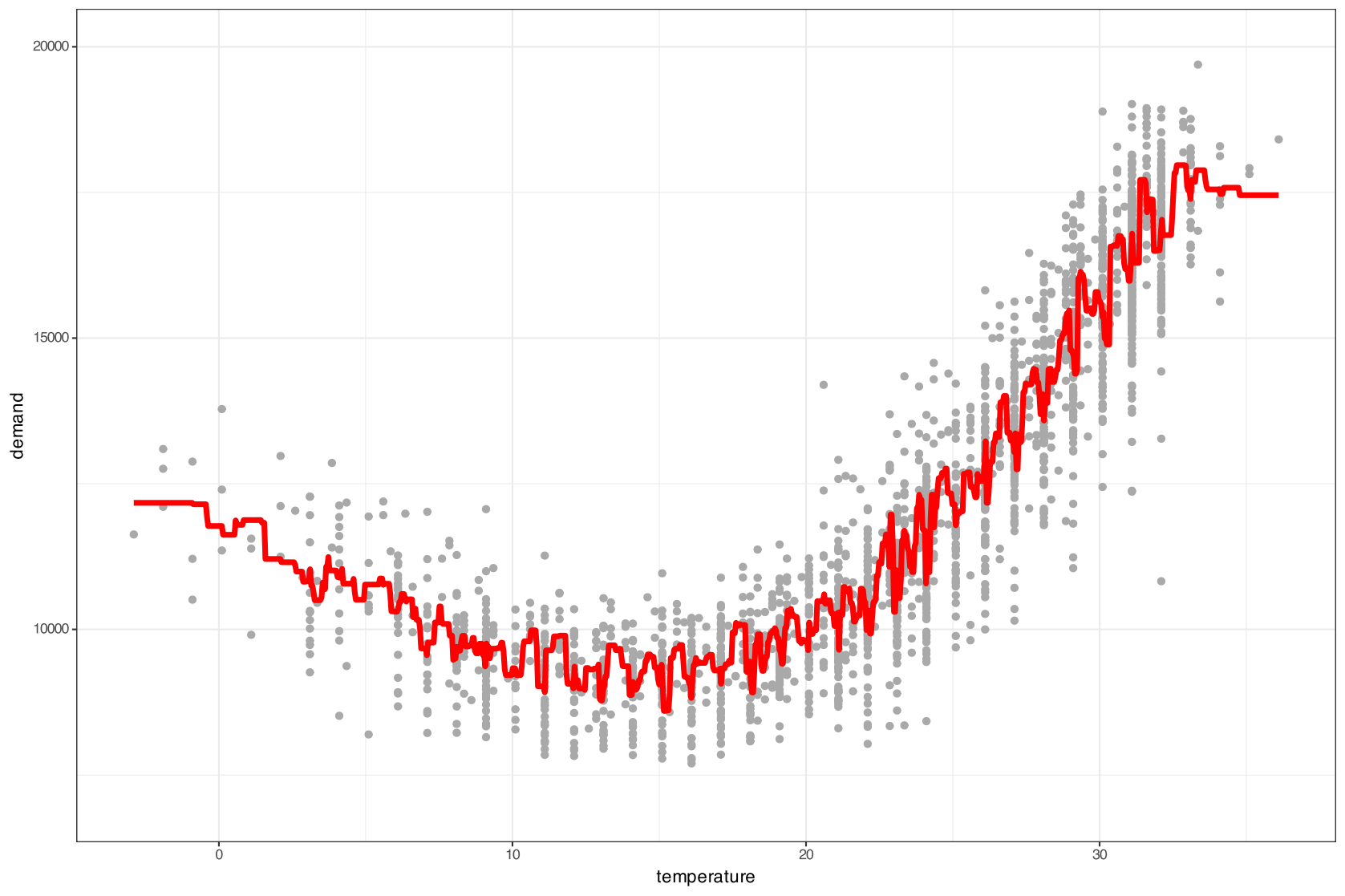
K=2



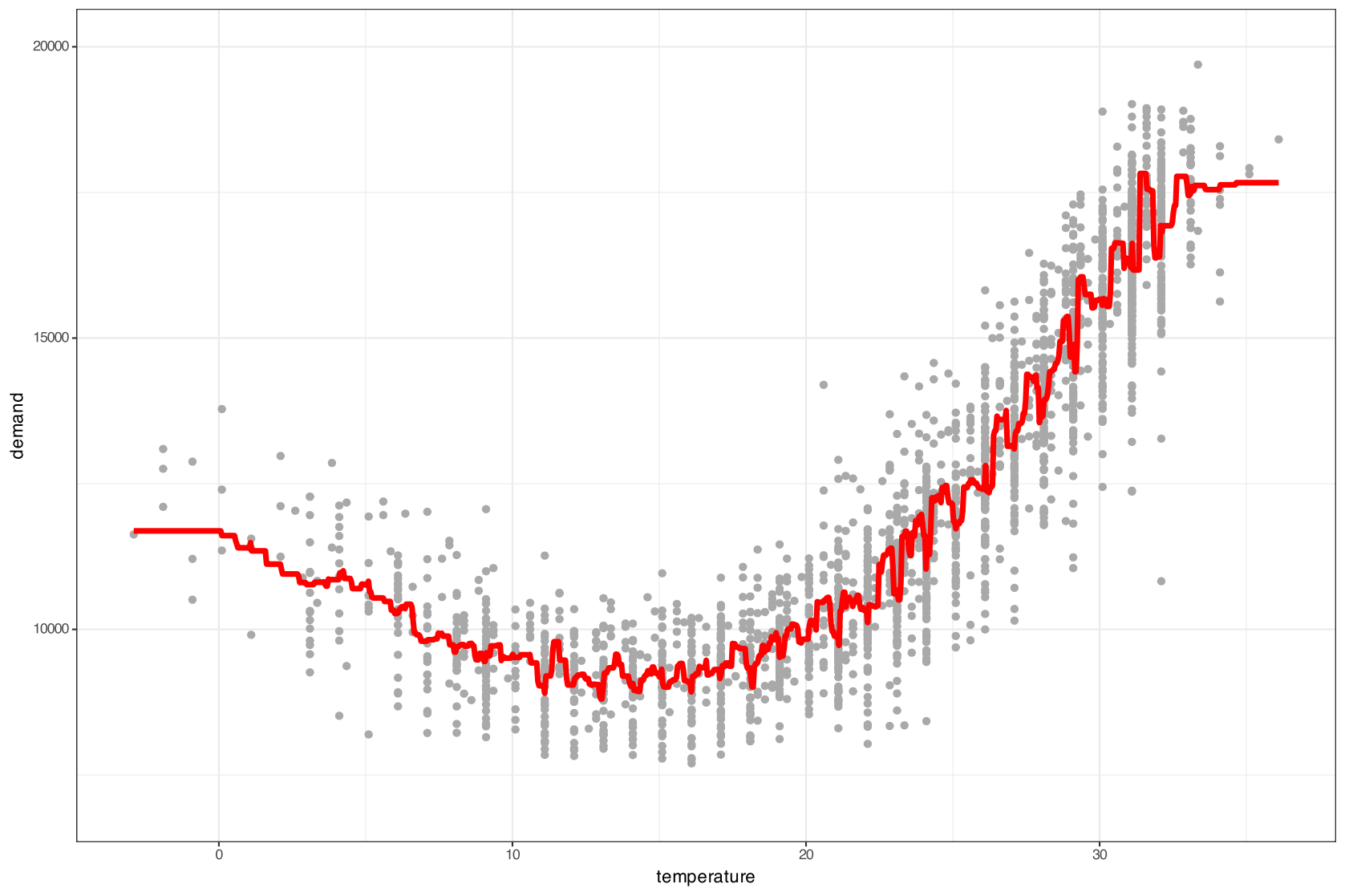
K=5



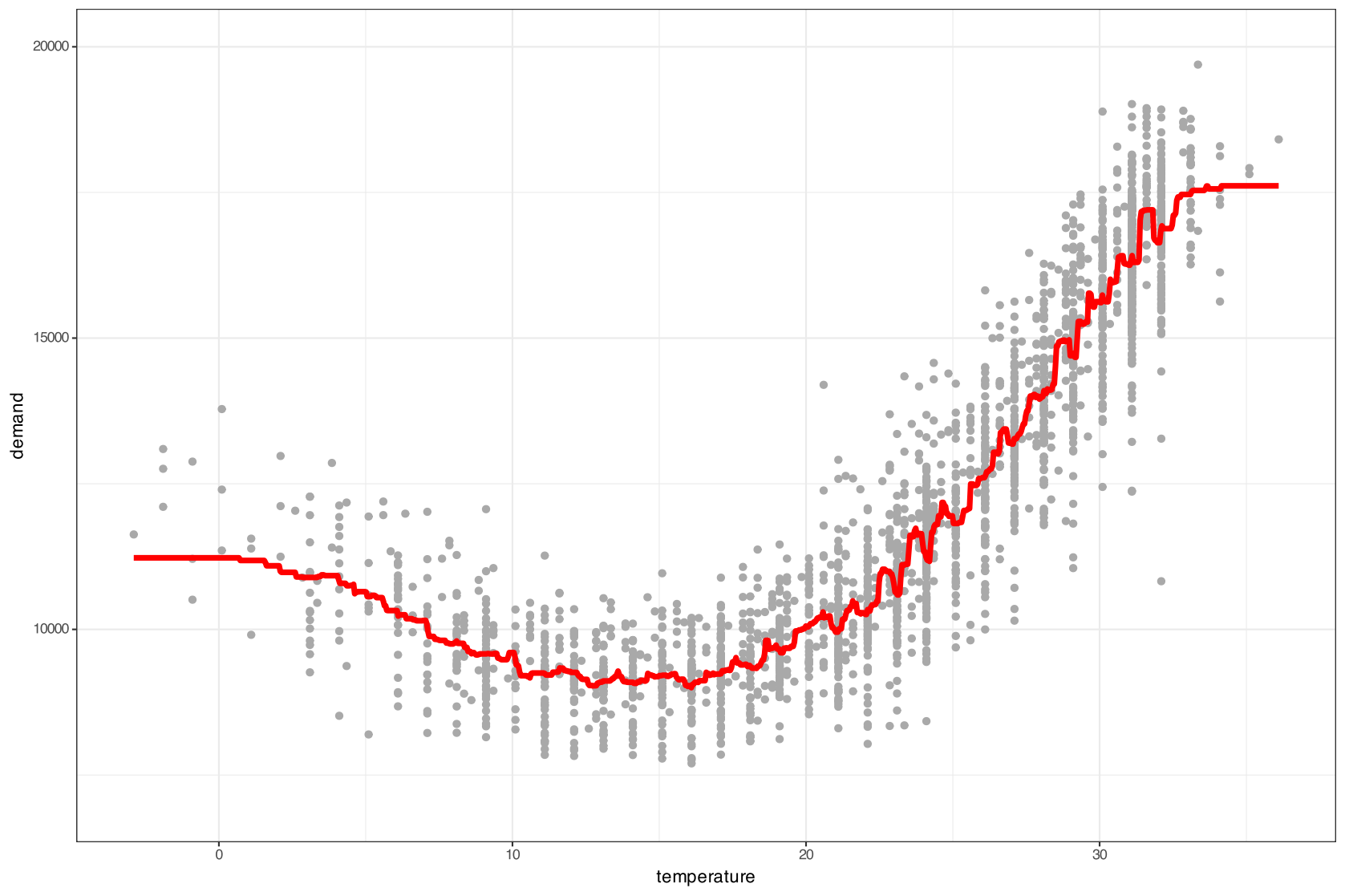
K=10



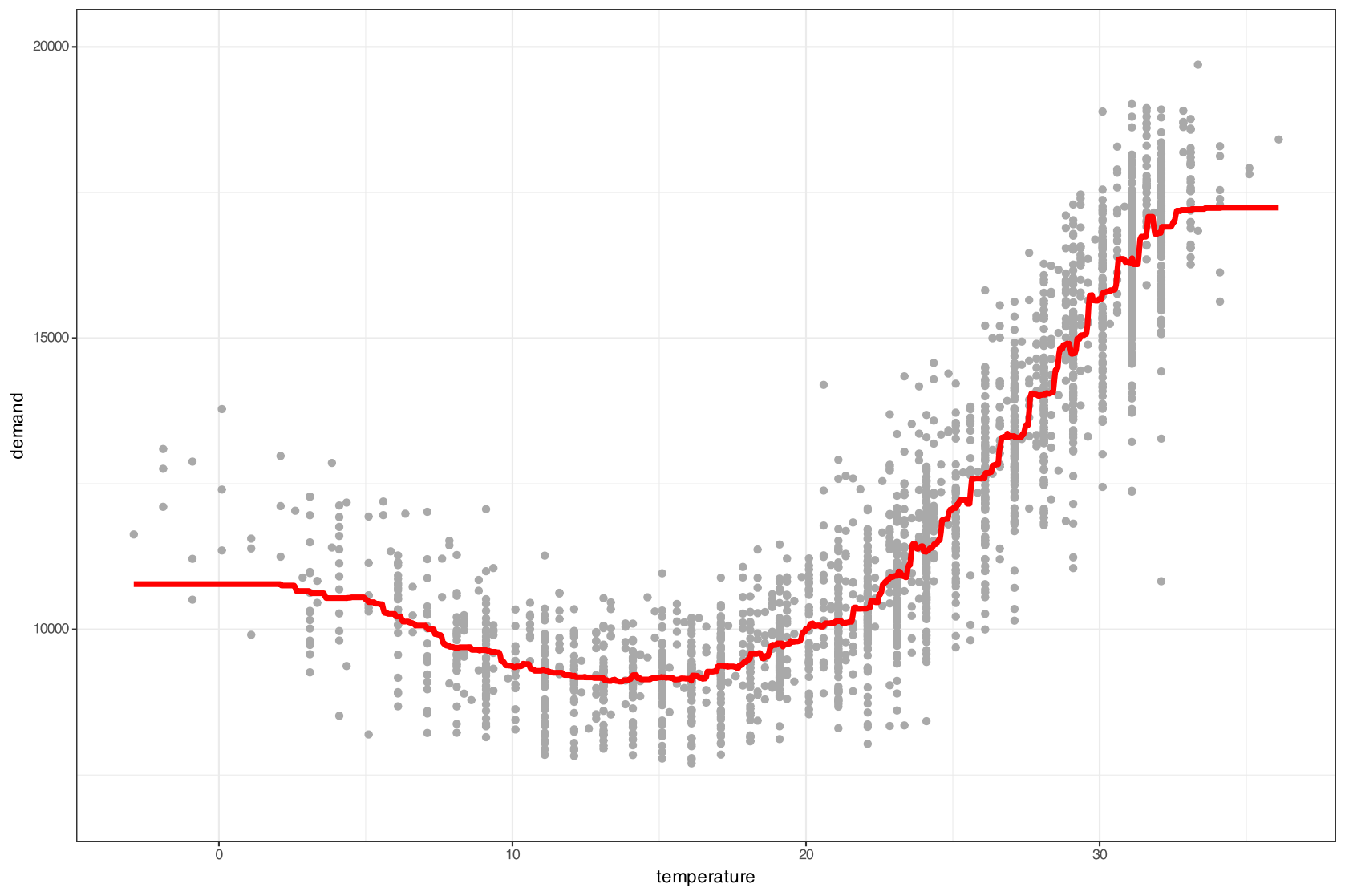
K=20



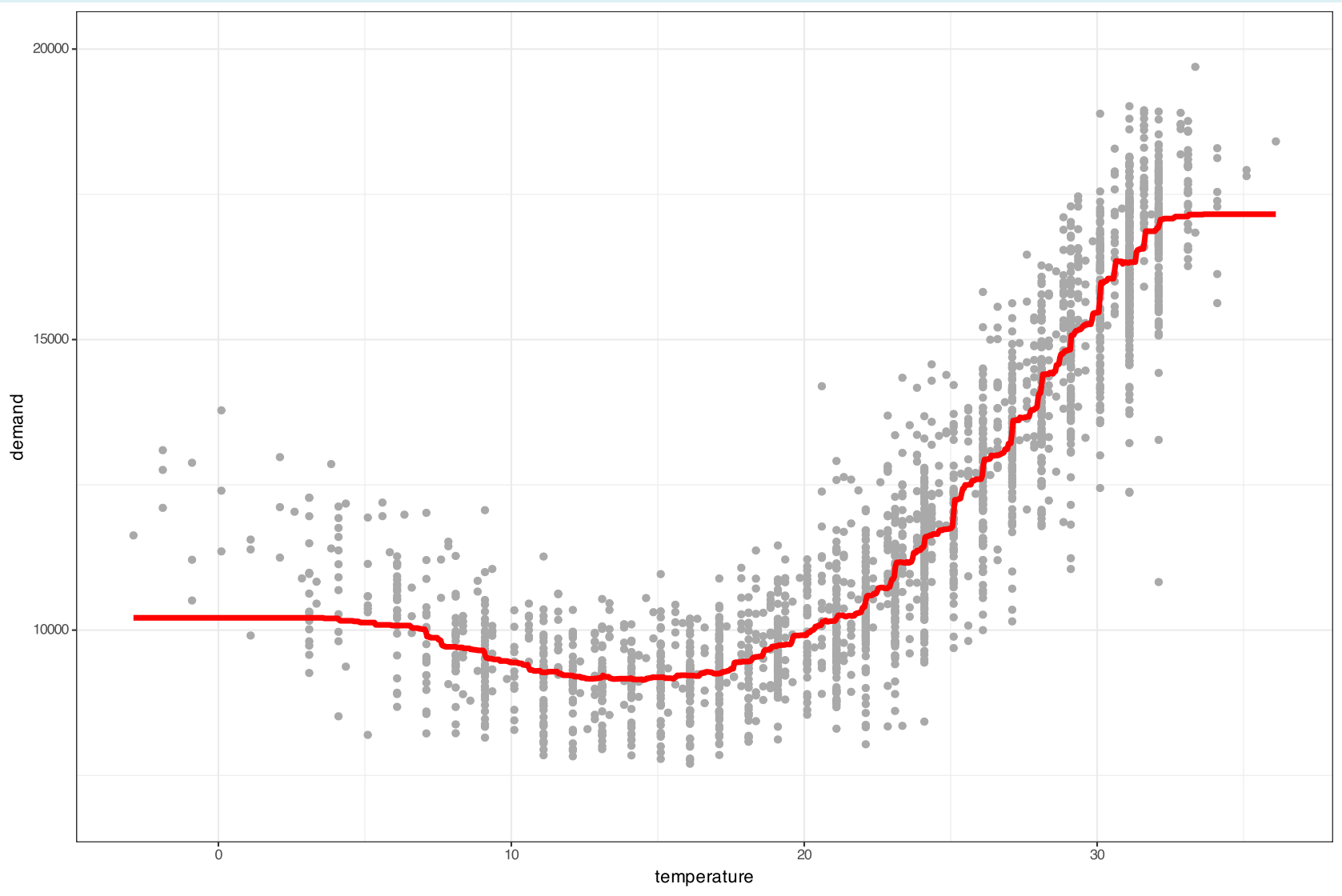
K=50



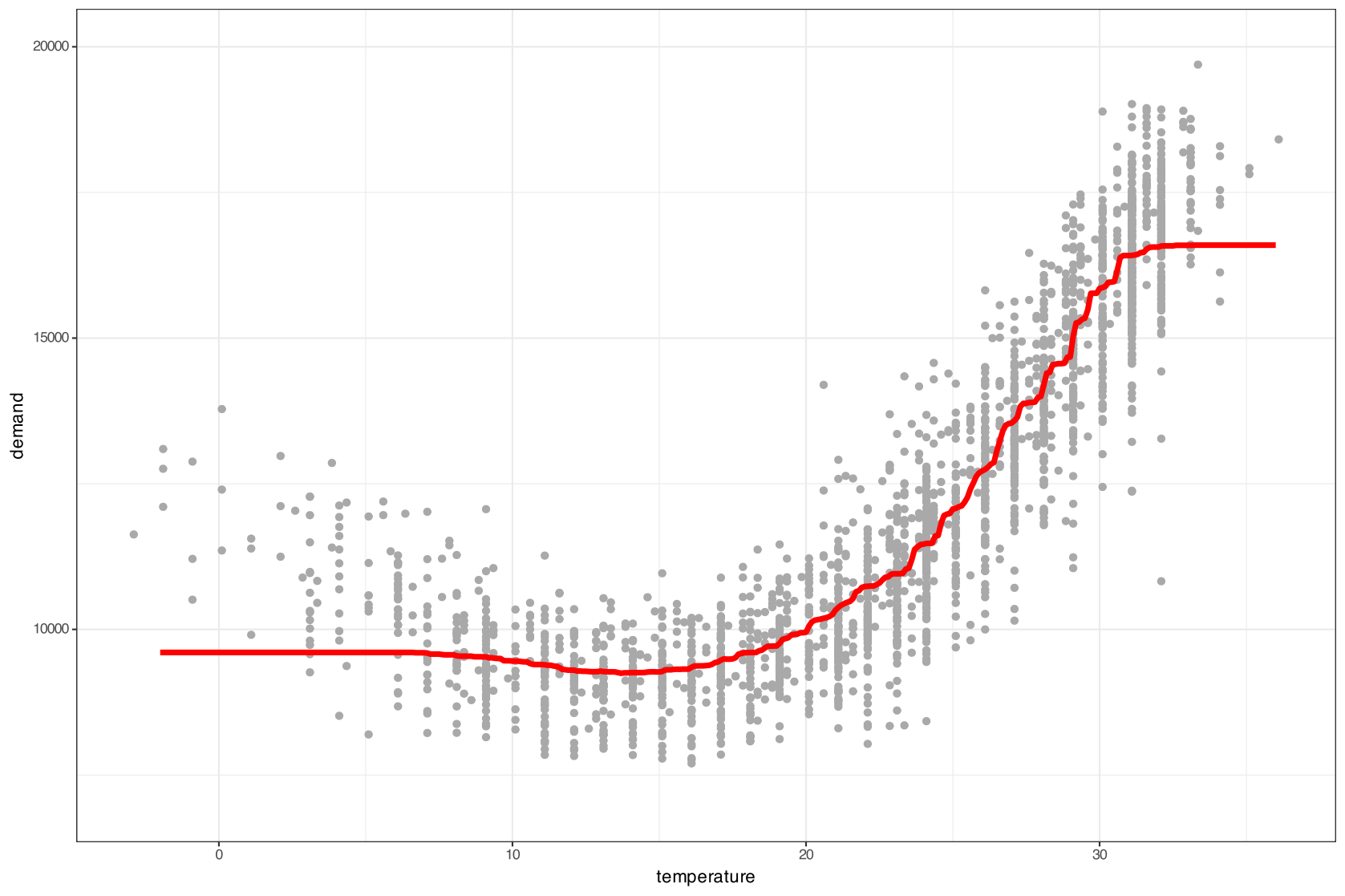
K=100



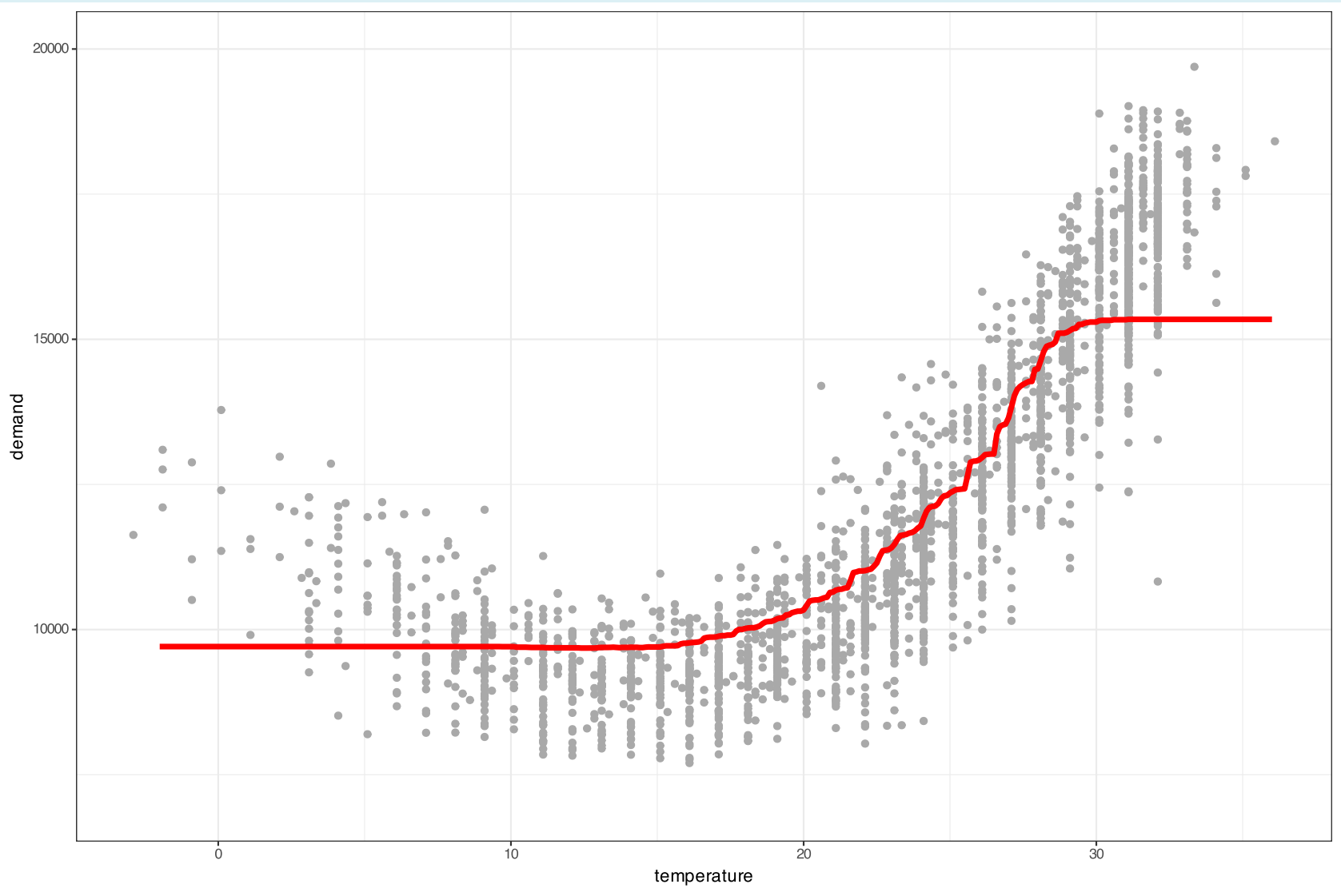
K=200



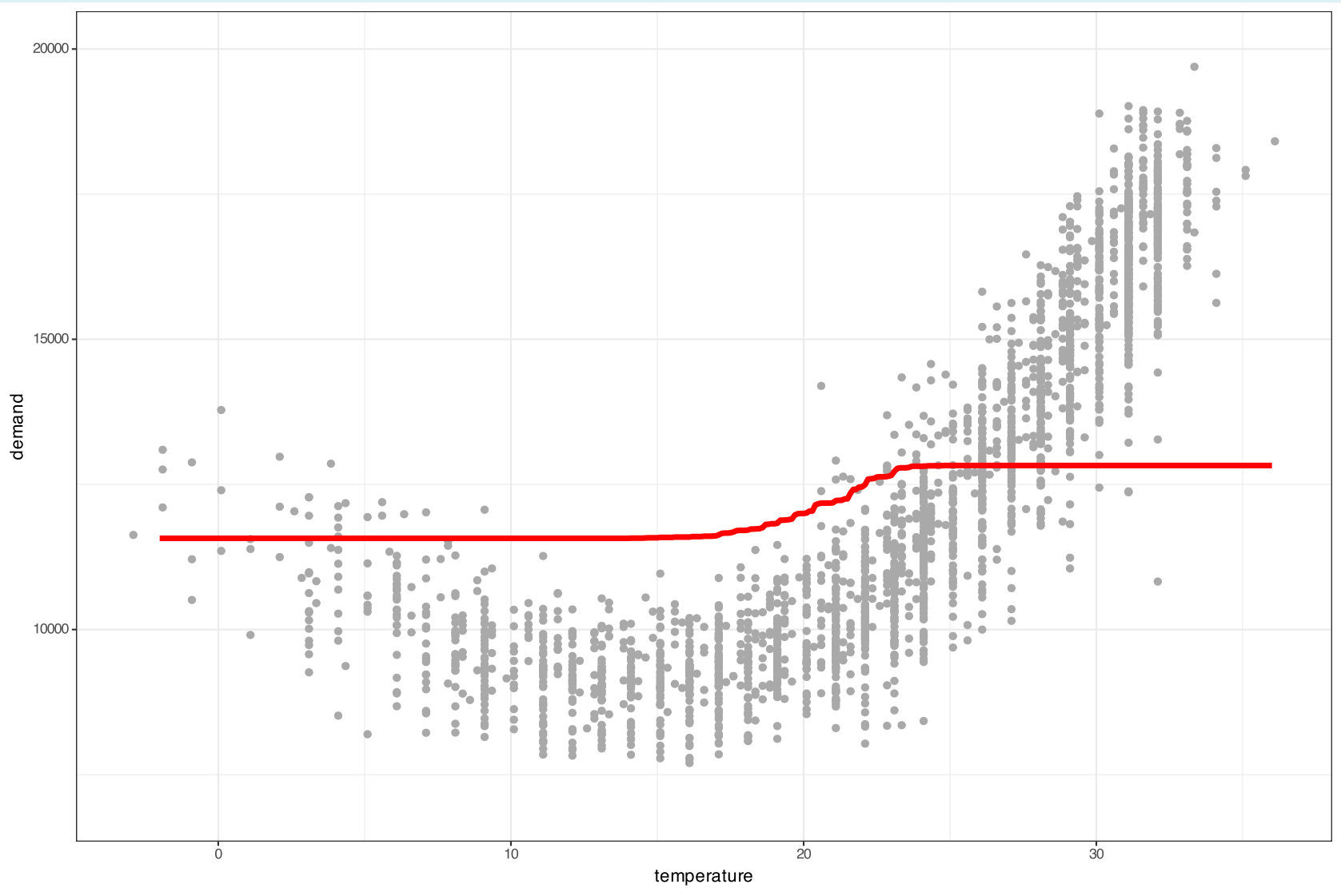
K=500



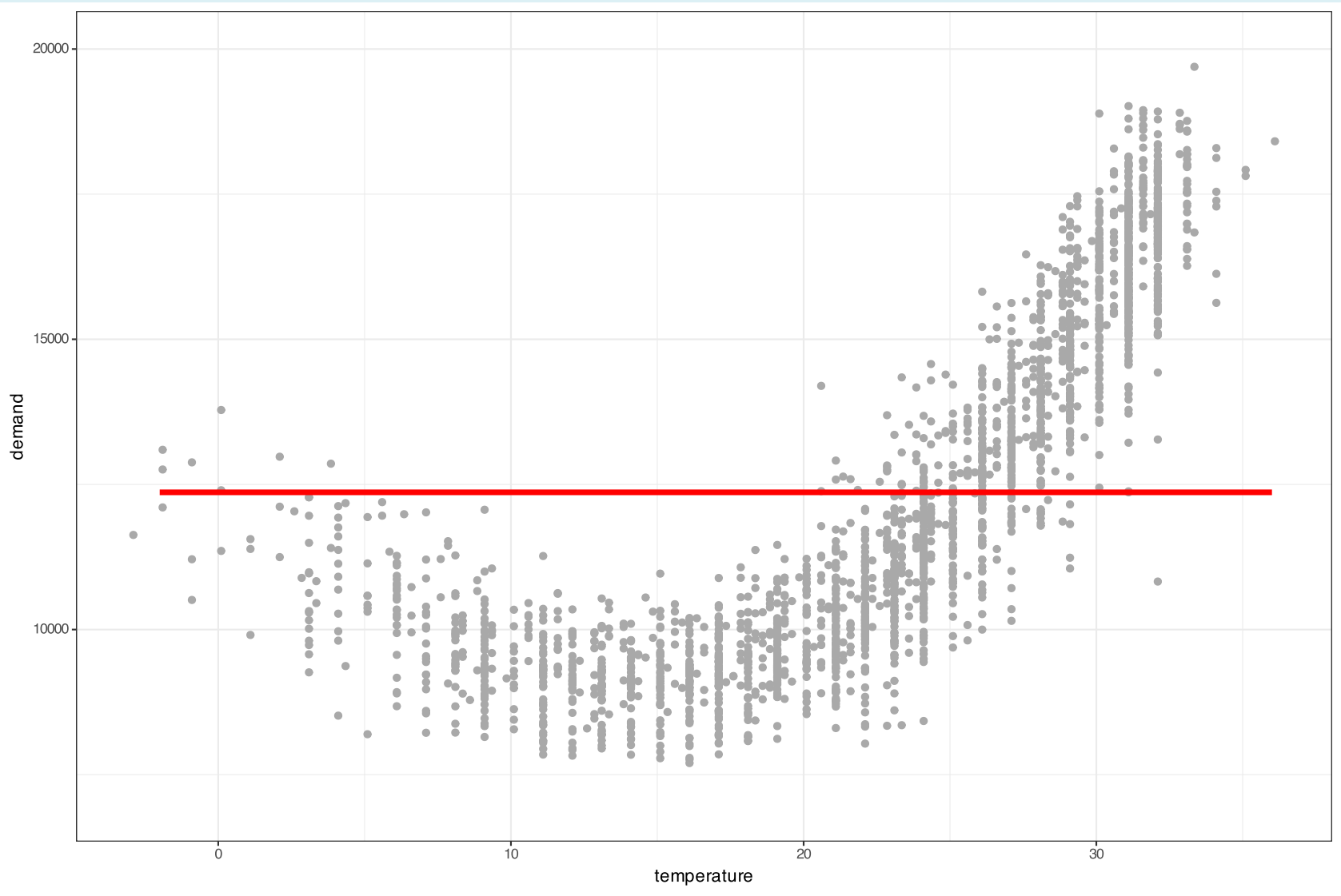
K=1000



K=2000



K=2357



Complexity and Generalization

- Smaller values of K give *more flexible*, but *less stable* function estimates:
  - they can capture very fine-scale structure in $f(x)$, because they're only averaging points from a small neighborhood...
  - but they can also confuse noise for signal!
- Larger values of K give *less flexible*, but *more stable* function estimates:
  - they can't adapt as much to wiggles in $f(x)$, because they're averaging points over a larger neighborhood.
  - but this makes them less prone to confusing noise for signal.
- There should be a **optimal medium** somewhere.
- **Question**: *How can we find it?*

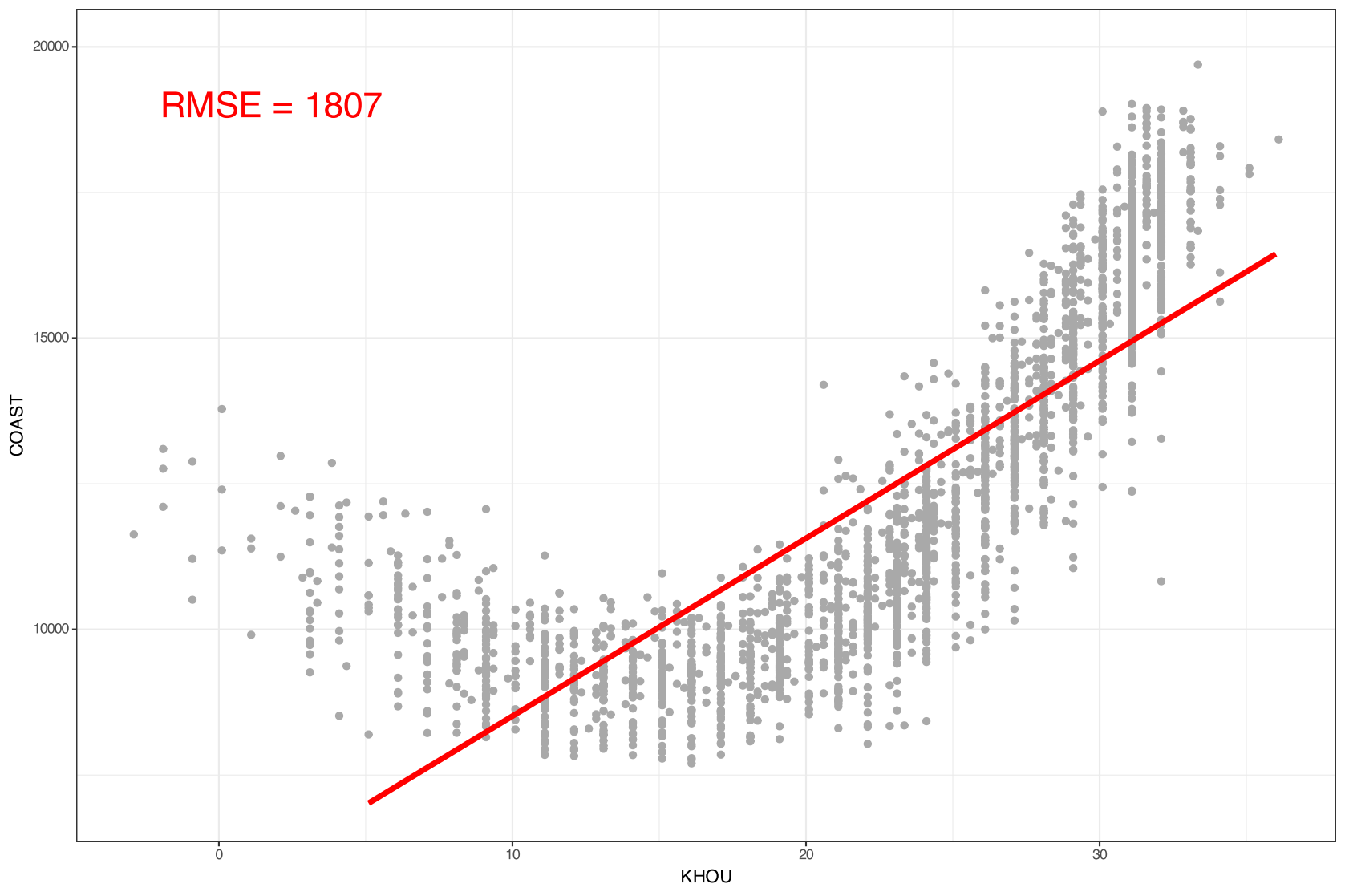
Measuring accuracy:EMSE

- **Answer**: Choose the model that makes the most accurate predictions, on average.
- Expected Mean Squared Error (EMSE)
- The sample version of EMSE is Root Mean Squared Error (RMSE)

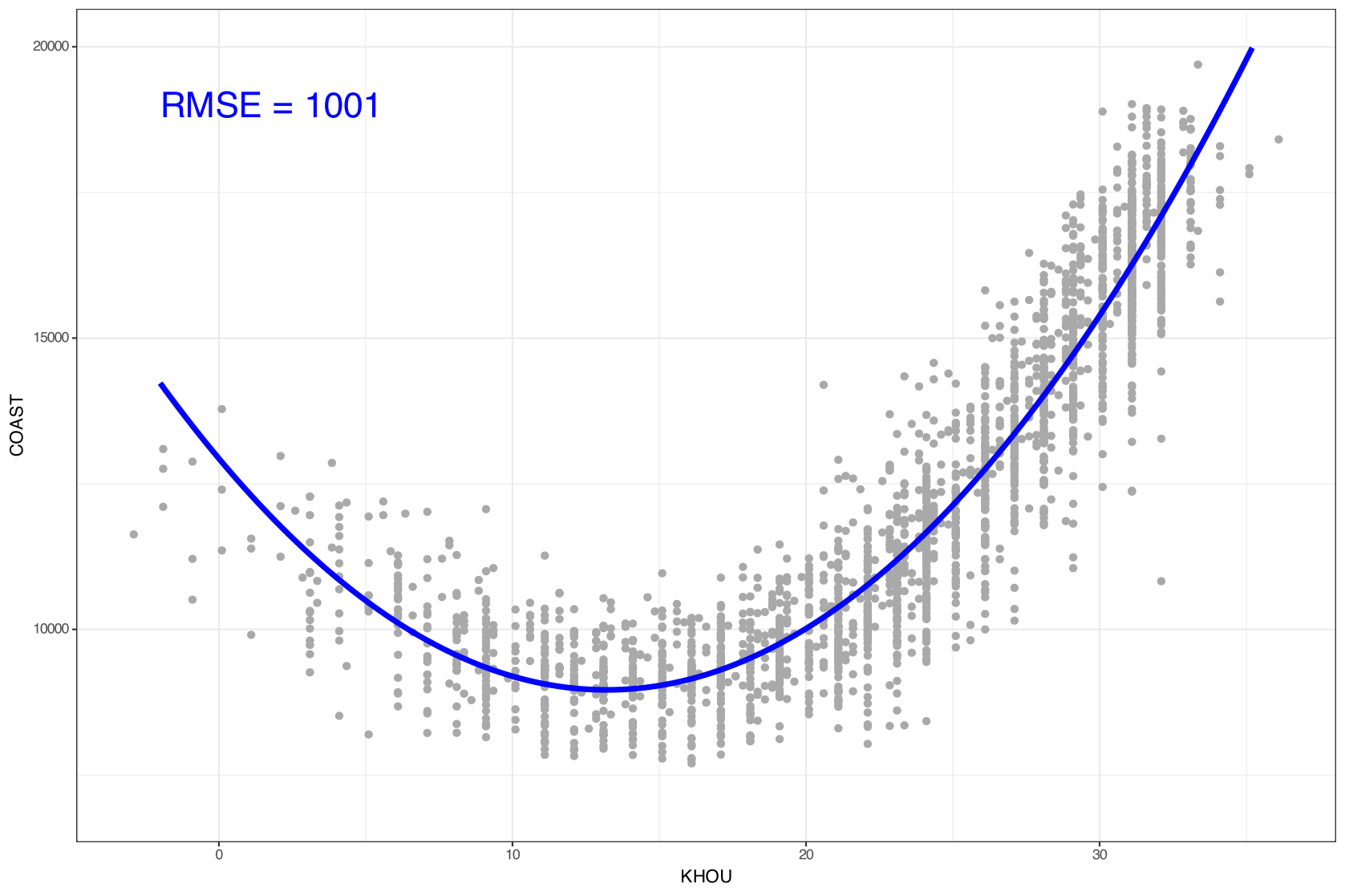
$$RMSE_{in} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2}$$

- This measures, on average, *how large are the errors made by the model on the training data*.
  - OLS minimizes this quantity over the set of linear functions.

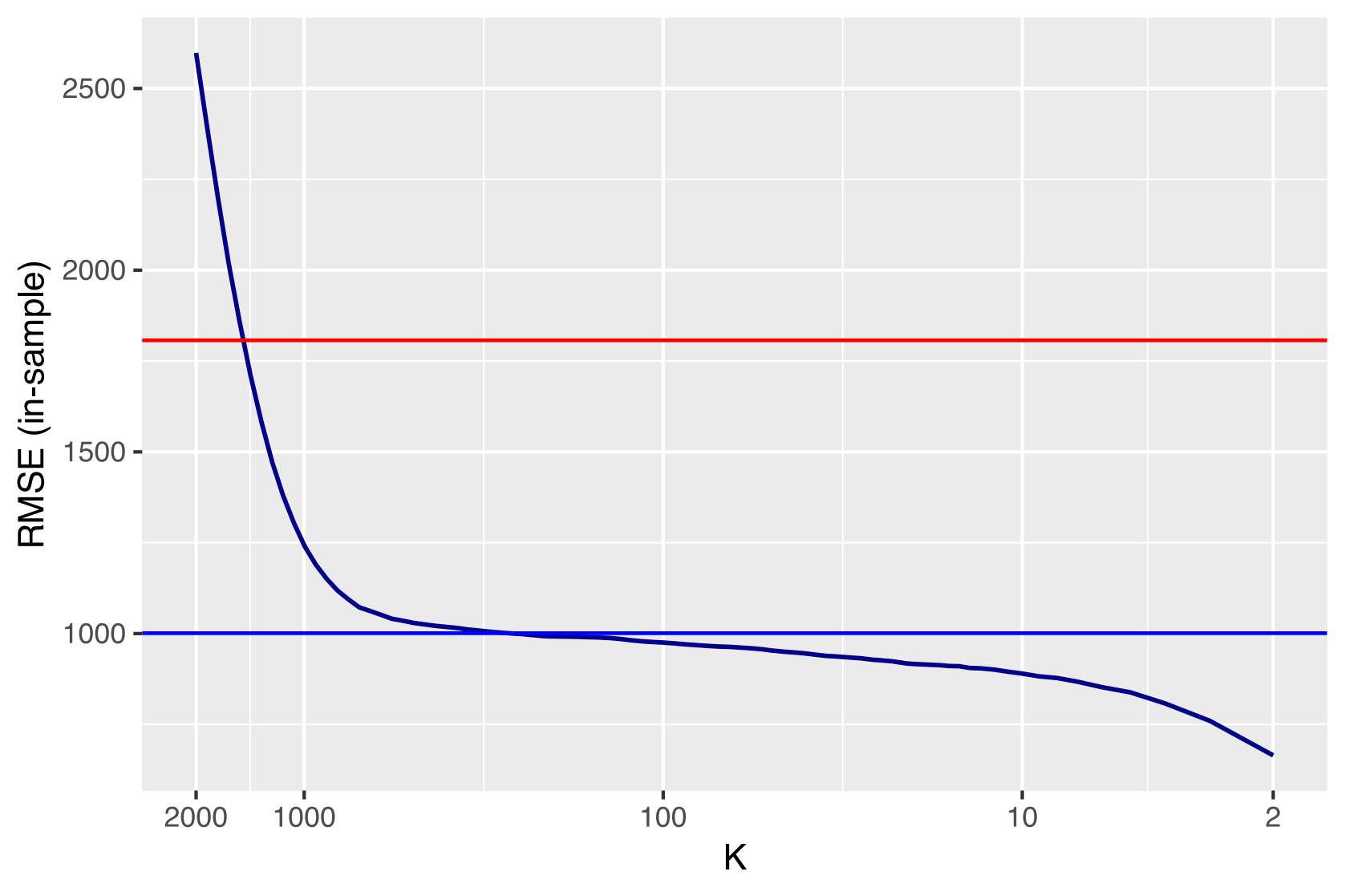
Measuring accuracy: linear vs. quadratric



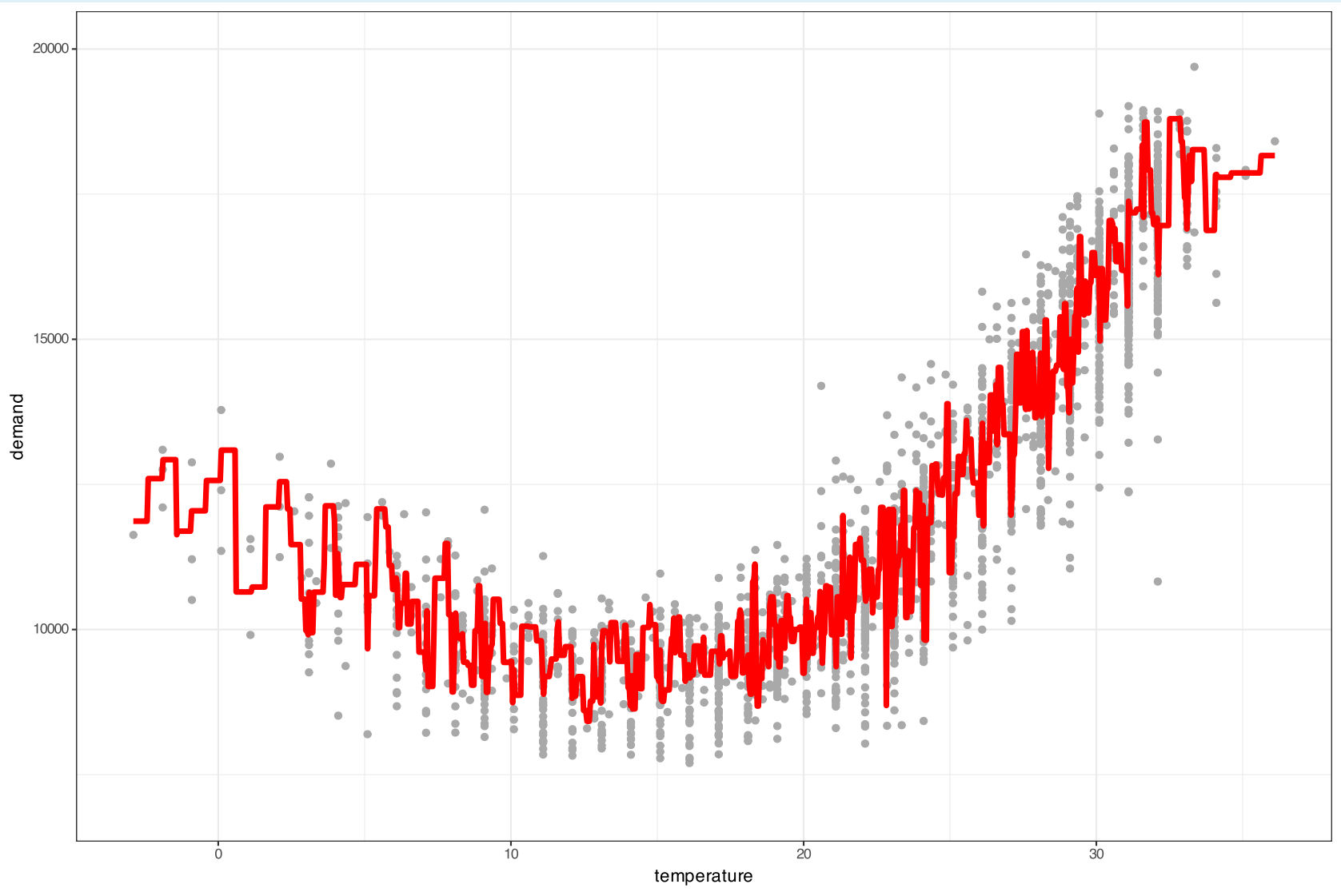
Measuring accuracy: linear vs. quadratric



Measuring accuracy: RMSE of K for KNN



So we should pick K=2?



Overfitting and Bias-Variance Trade-off

Overfitting

- Actually, it is not a good idea to fit the model too well like K=2.
  - Empirically, it tends to lead to a terrible prediction.
- Why?
  - Because the model is too flexible, it tends to absorb all the idiosyncratic noise(ϵ) in the prediction model.
  - A new observation with the same X will have a different idiosyncratic noise, and so the prediction is off.
  - Remember: *our aim is not to fit the model but to predict future*
- To avoid overfitting, we need to find the optimal degree of flexibility

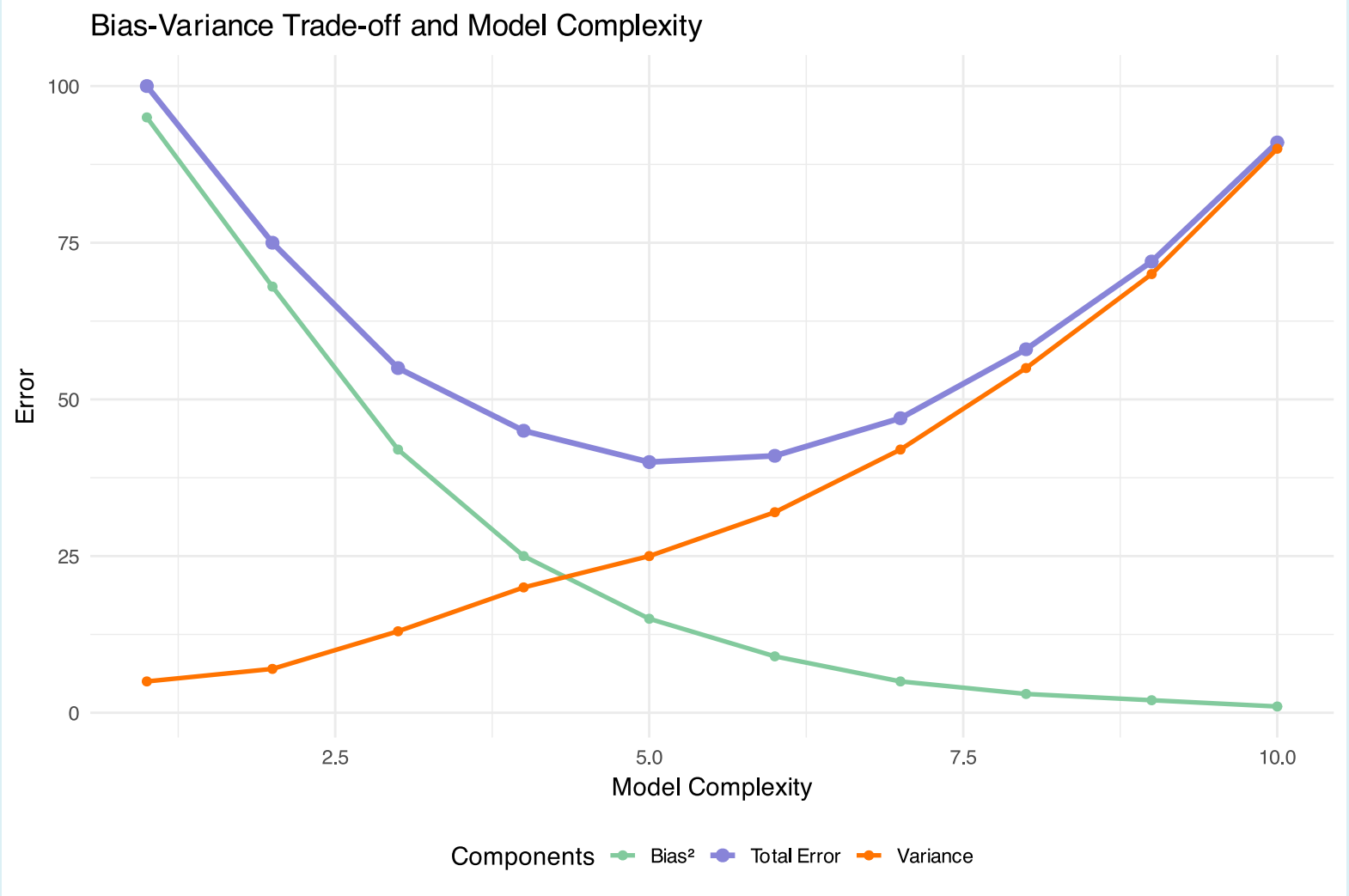
Bias-variance trade-off

- The expected squared error can be decomposed into:

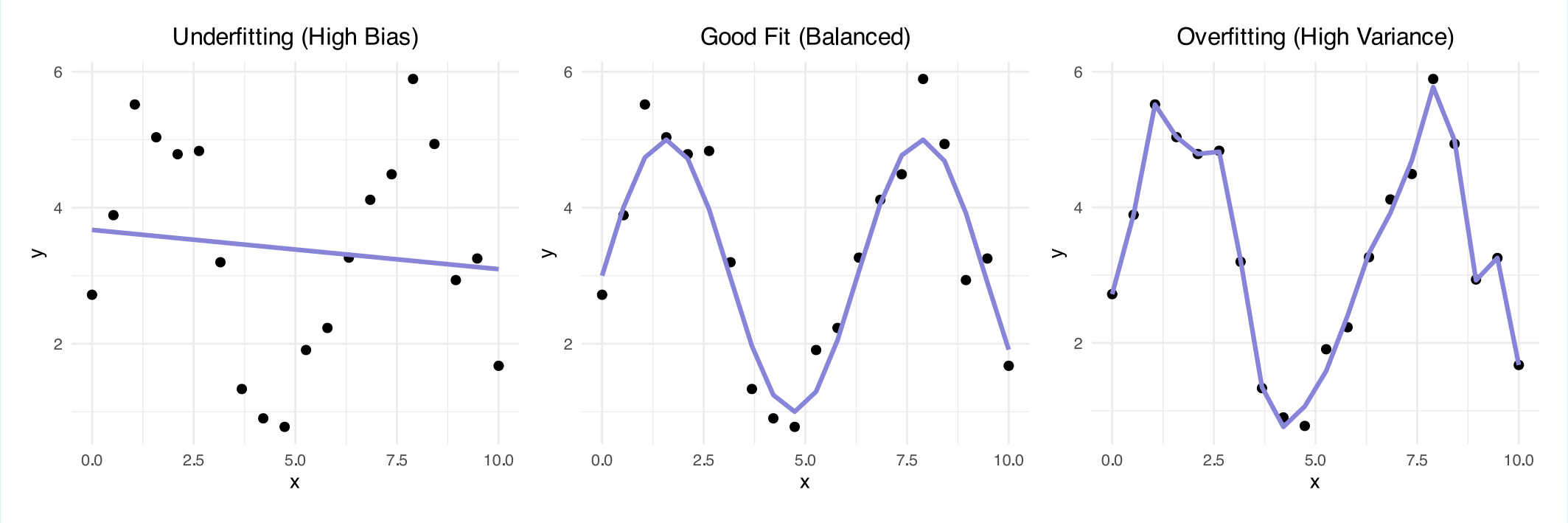
$$EMSE = E[(Y - \hat{Y})^2] = \underbrace{[f(X) - E(\hat{f}(X)]^2}_{\text{Bias}^2} + \underbrace{E[(\hat{f}(X) - E(\hat{f}(X))]^2}_{\text{Variance}} + \underbrace{E[\epsilon^2]}_{\text{Noise}}$$

- $f(x)$ is the true function
- $\hat{f}(x)$ is the model prediction
- σ_ϵ^2 is the inherent noise in the data (cannot be eliminated)
- Bias: Systematic error caused by approximating a complex general function by a restricted functional form.
- Variance: refers to the degree by which $\hat{f}$ would change if we estimated on a different data set.
- Total Error: Bias² + Variance + Irreducible Error (noise)

Total Error vs Model Complexity



Visualization of Three Fitting Types



- Underfitting: High Bias, Low Variance
- Overfitting: Low Bias, High Variance

--

- Good Fit: Balanced

Bias-variance trade-off

- High K = high bias, low variance:
  - Estimate $f(x)$ using many points, some of which might be far away from x. These far-away points bias the prediction; their values of $f(x)$ are slightly off on average.
  - But more data points means lower variance: less chance of memorizing random noise.
- Low K = low bias, high variance:
  - Estimate $f(x)$ using only points that are *very close* to x. Far-away x points don't bias the prediction with their "slightly off" y values.
  - But fewer data points means higher variance: more chance of memorizing random noise.
- Question: Why $K = 2$ minimizes the RMSE?

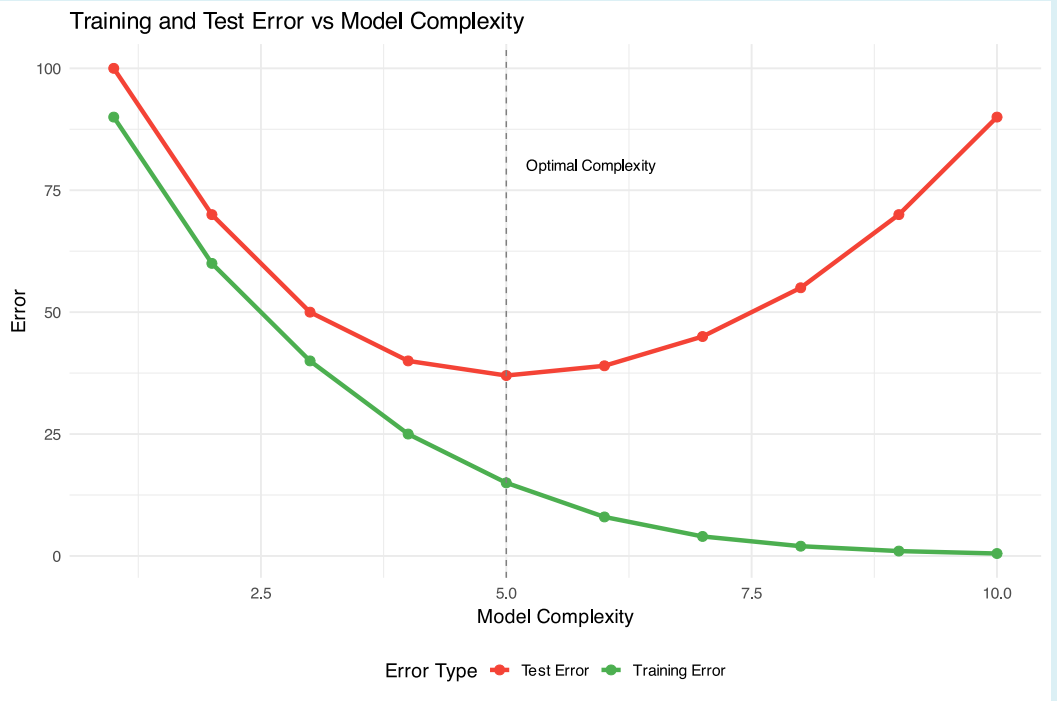
Out-of-sample vs in sample

- Answer: K=2 model earned a low RMSE by simply memorizing the random pattern of noise in the data in the past.
- However, our object is to make a better prediction.
- Therefore we divide the whole sample into two subsets
  - **In sample** or training data: to fit the model
  - **Out-of-sample** or testing data: Additional data used to evaluate how good is the regression model fit(assume "future")
- Suppose we have data $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n), (x_{n+1}, y_{n+1}) \dots (x_{n+m}, y_{n+m})$
  - n in sample: $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$
  - m out-of-sample: $(x_{n+1}, y_{n+1}) \dots (x_{n+m}, y_{n+m})$

Out-of-sample accuracy

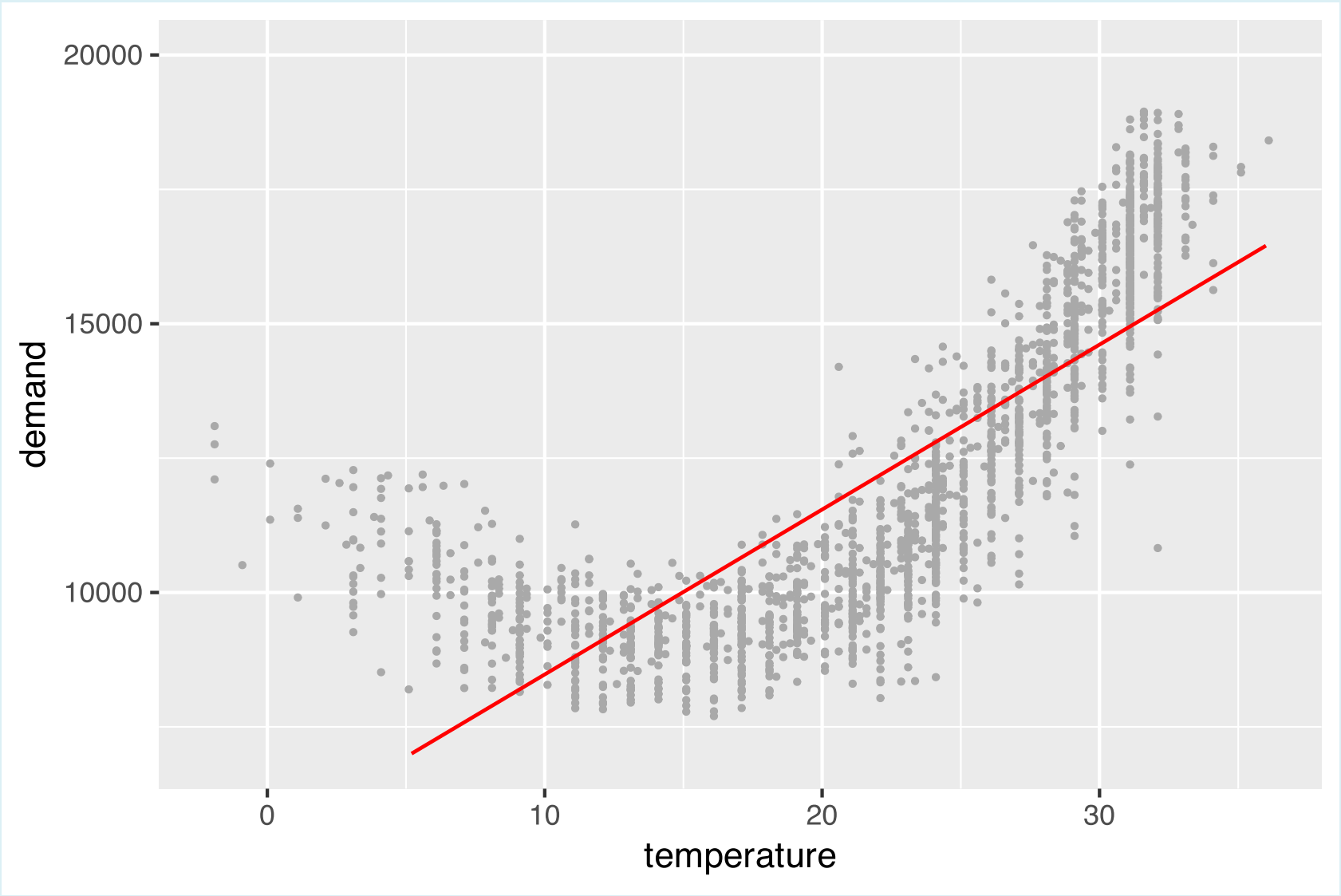
- Key idea: what really matters is our prediction accuracy out-of-sample!
- Therefore, we only care about the $RMSE$ of out-of-sample instead of *in sample*.

$$RMSE_{out} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2}$$

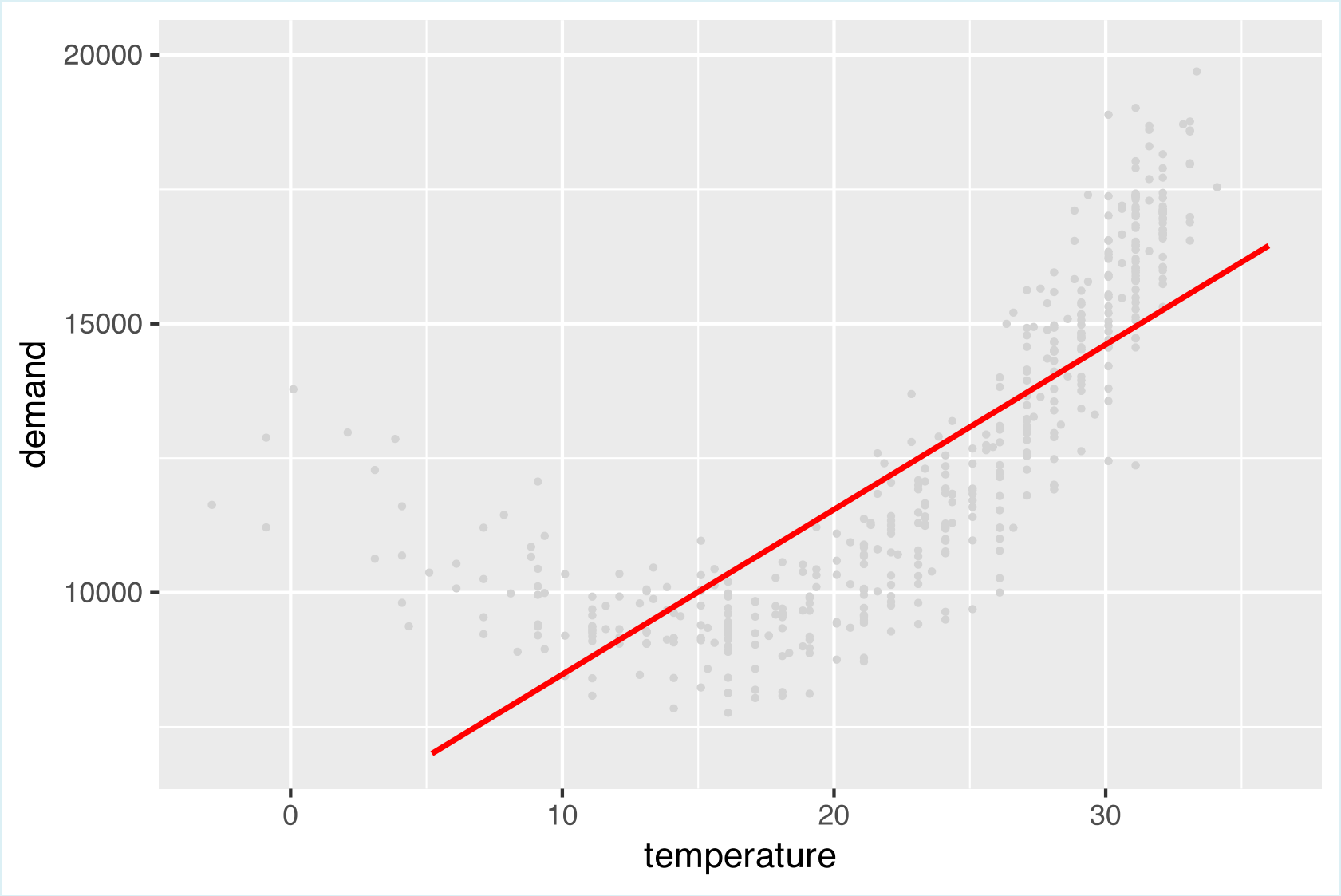


- The optimal model complexity is the one that minimizes the test error

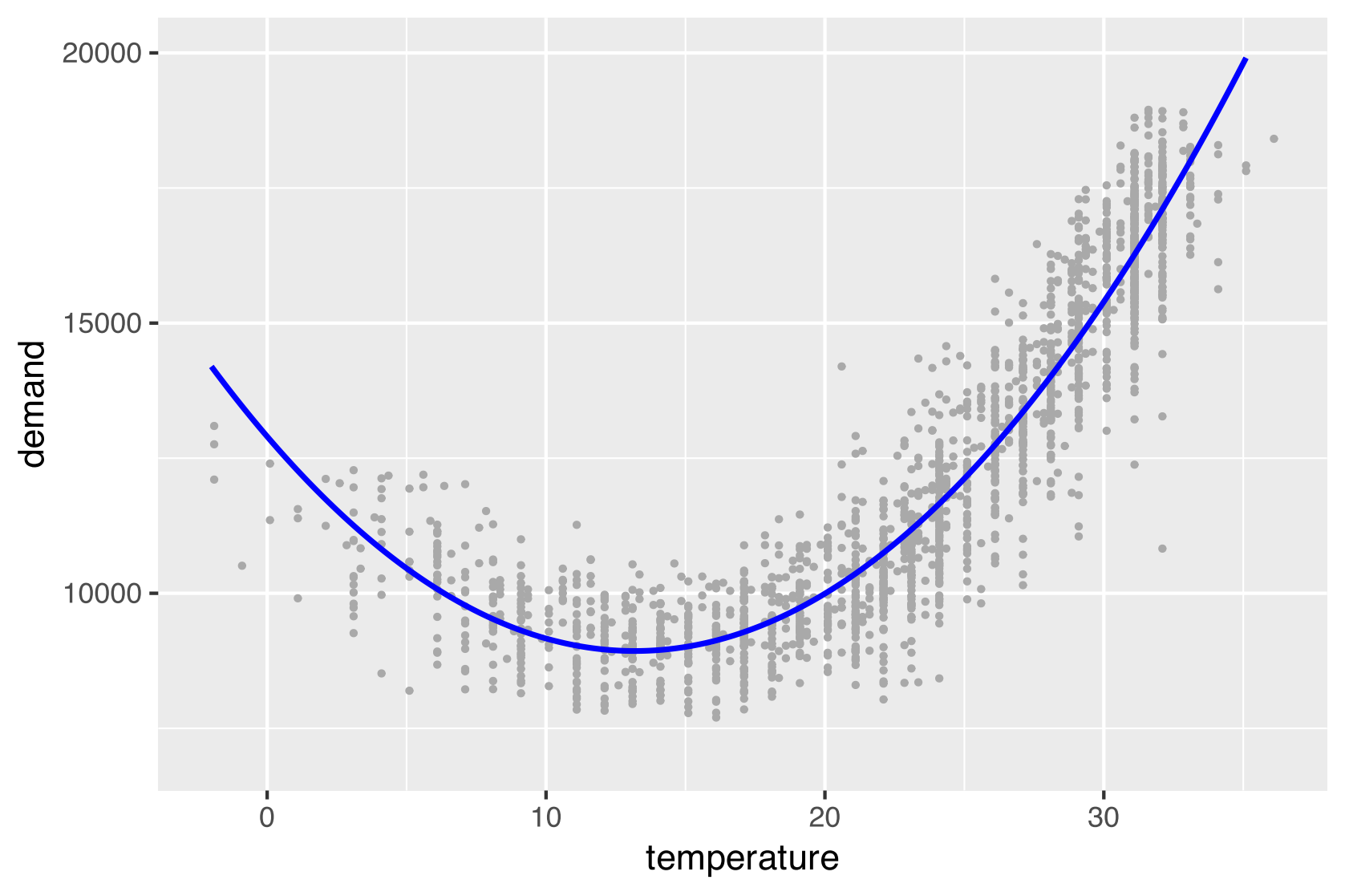
Linear model: train



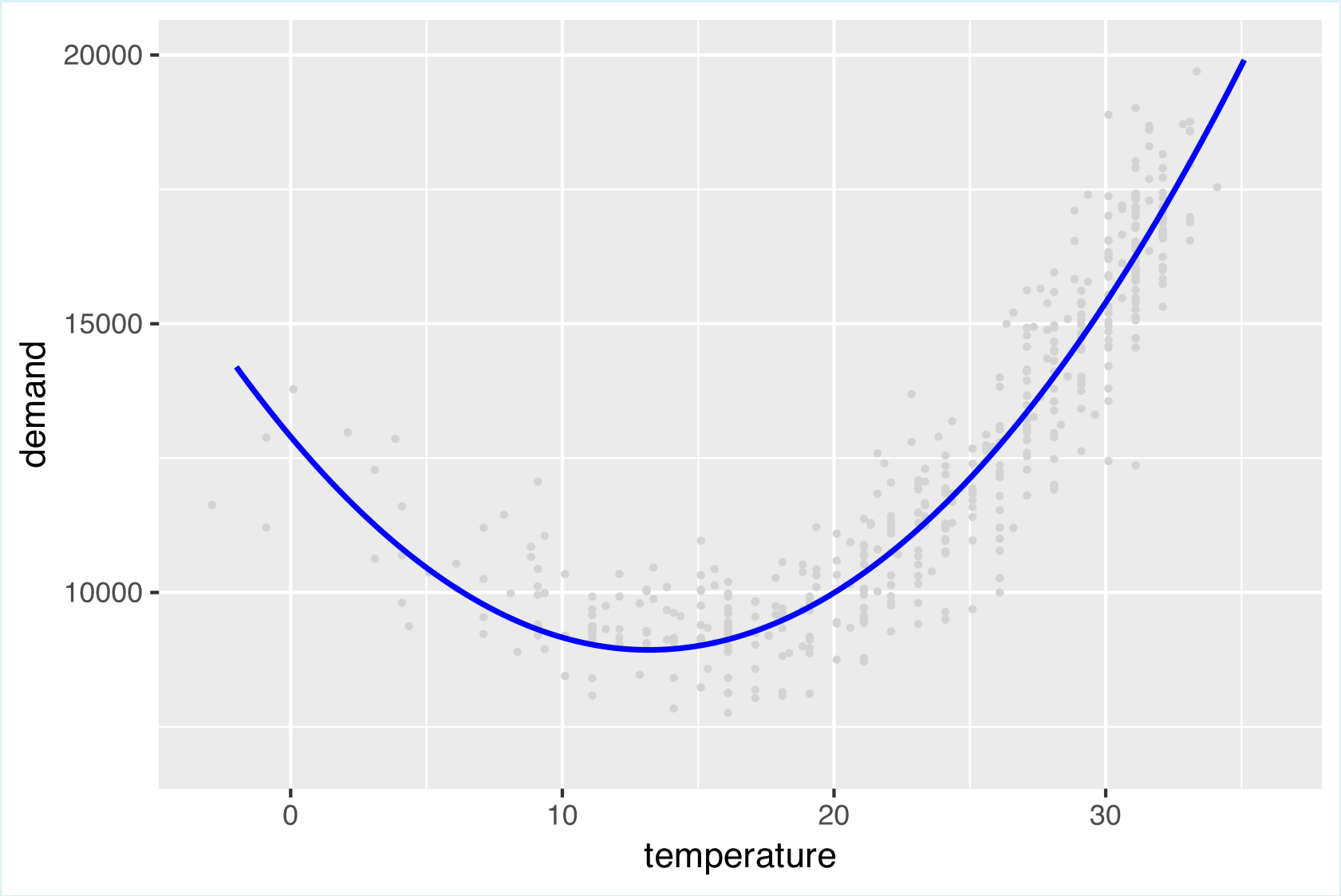
Linear model: test



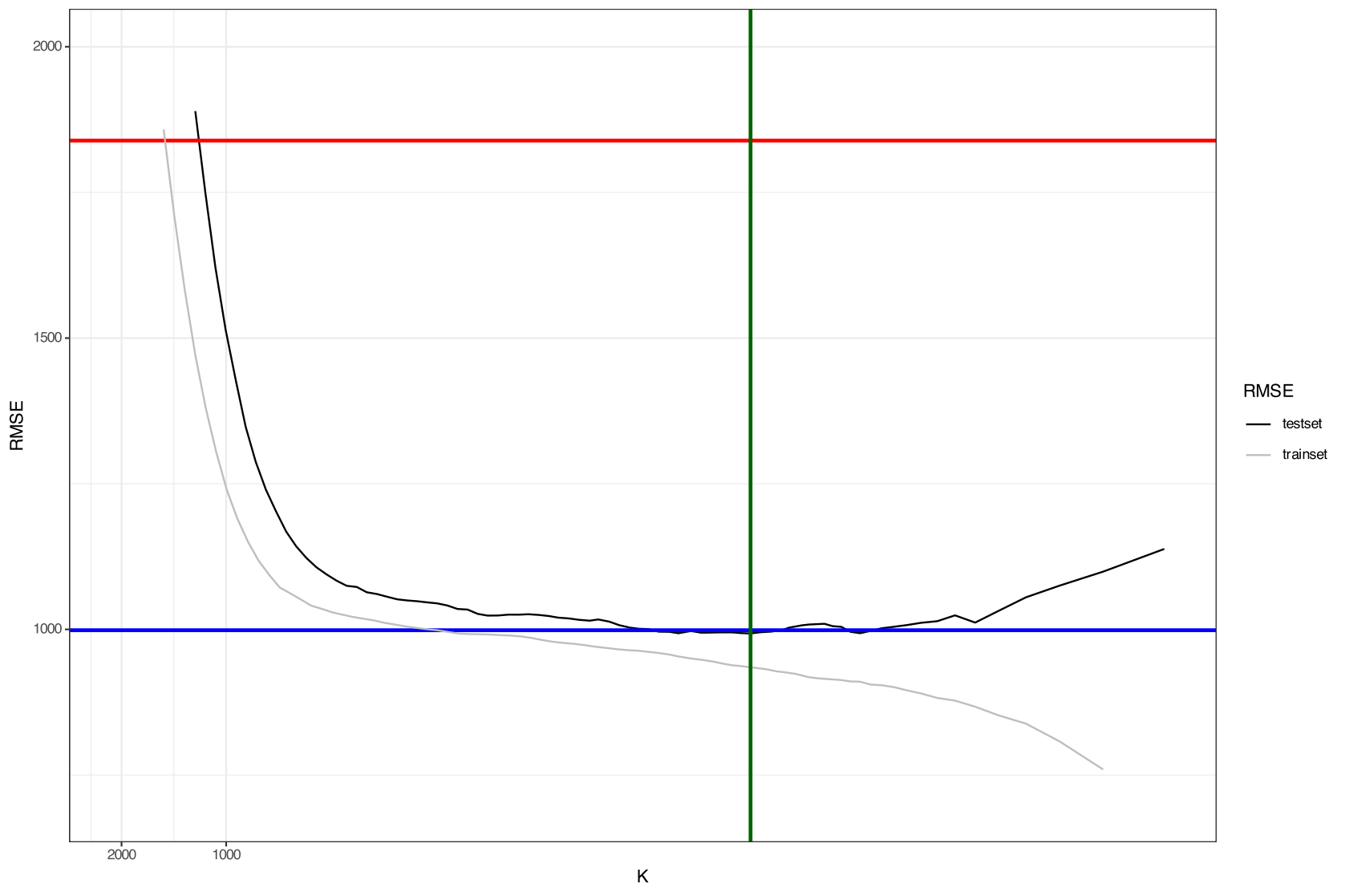
Quadratic model: train



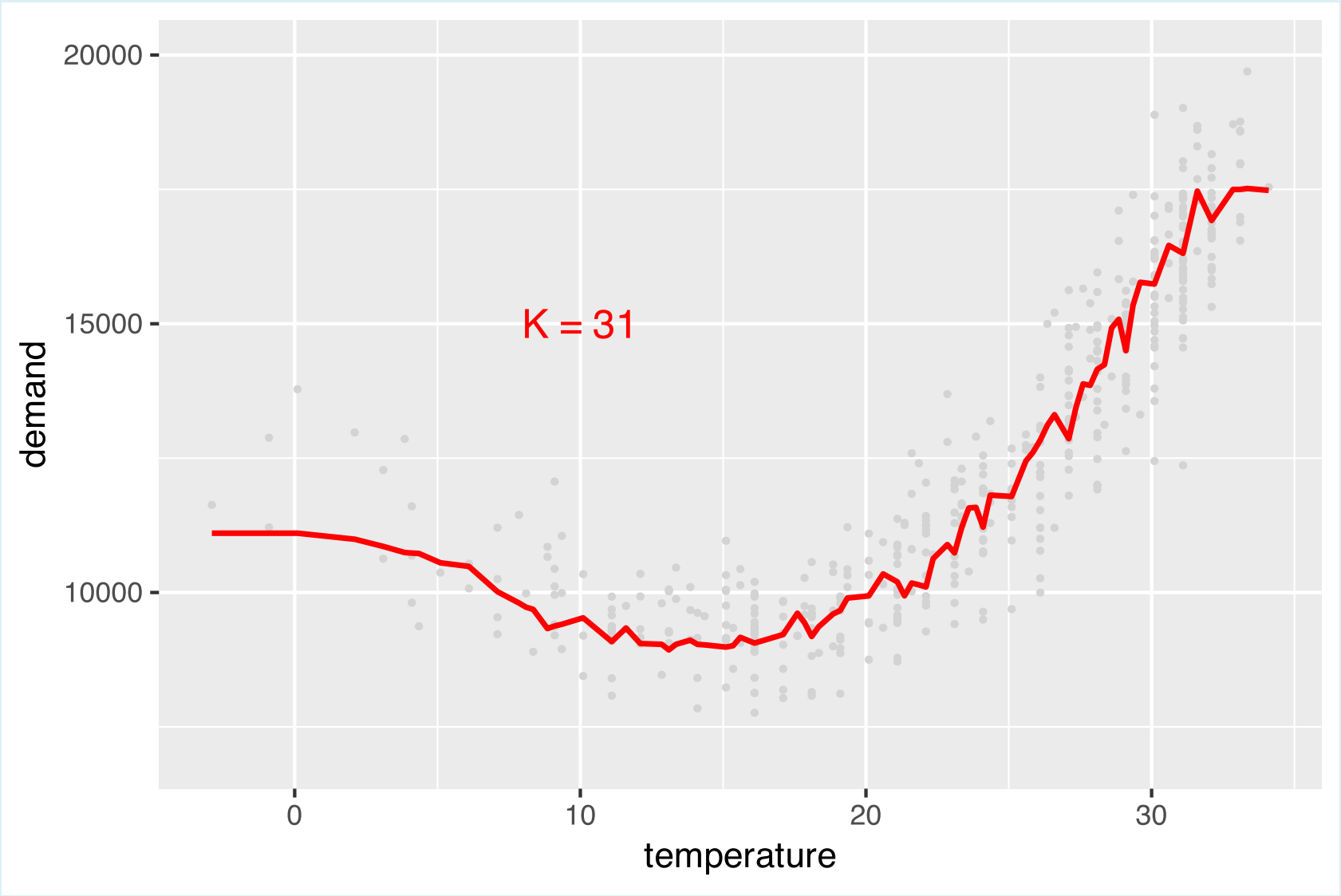
Quadratic model: test



K-nearest neighbors: test



K-nearest neighbors: test at the optimal k



RMSE: Linear v.s Quadratic vs.KNN

- Linear:

$$RMSE_{out} = 1839$$

- Quadratic

$$RMSE_{out} = 999$$

- K-Nearest Neighbors

$$\text{RMSE}_{out} = 993$$

- KNN is the best model in terms of out-of-sample accuracy.

Measuring model accuracy, revisited

- Recall that out-of-sample EMSE is defined as:

$$\text{EMSE}_{out} = E\left[\left(Y - \hat{f}\left(X\right)\right)^2\right]$$

- But in reality, in-sample and out-of-sample are not always the same.
  - In other words, out-of-sample EMSE is kind of a random sampling from the population.
- To estimate out-of-sample EMSE, we train our model $\hat{f}$ on in-sample data and calculate average performance on out-of-sample data:

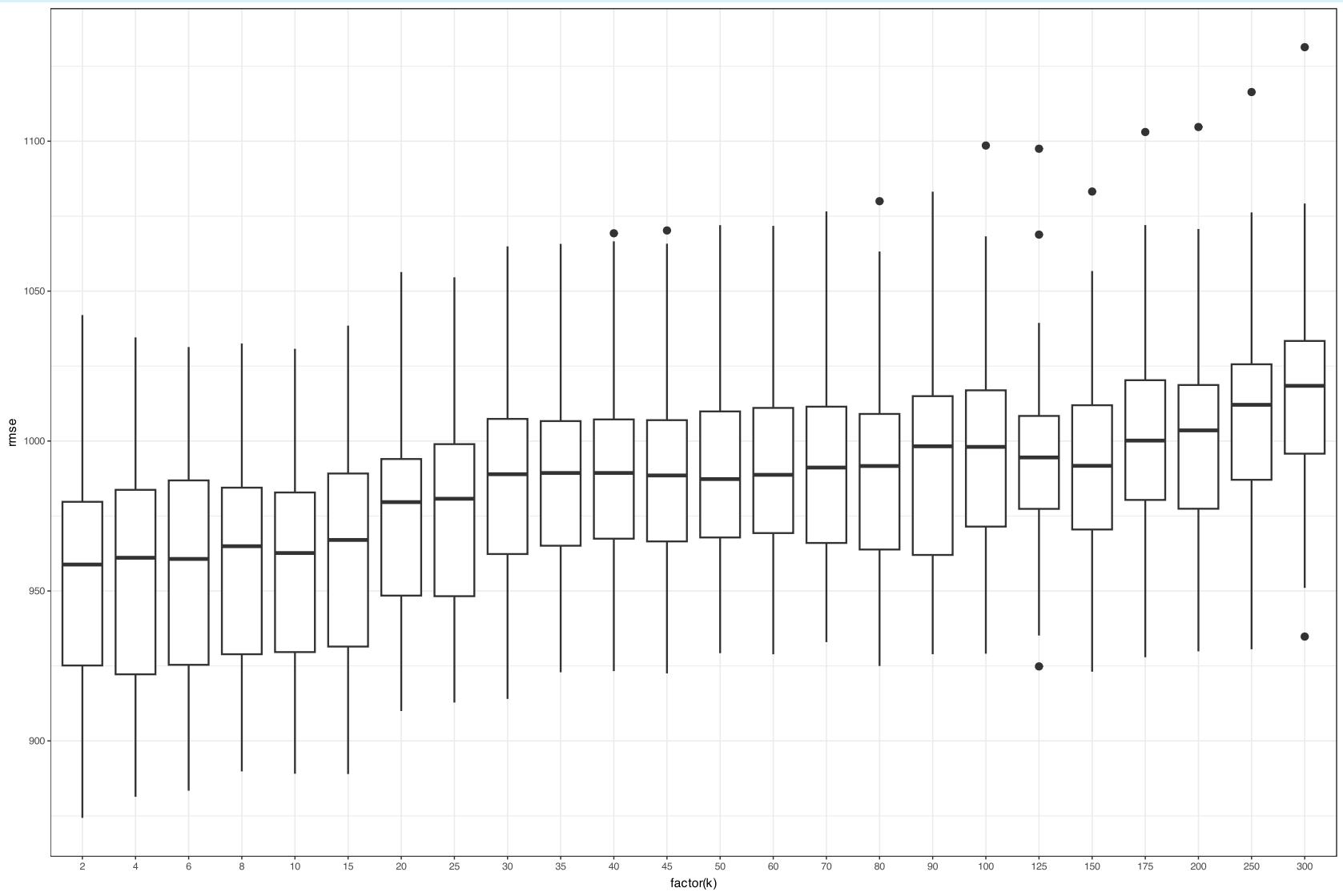
$$\widehat{EMSE} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} \left(y_i - \hat{f}\left(x_i\right)\right)^2$$

- This estimate has two sources of randomness:
  - The function estimate $\hat{f}\left(x\right)$ from in-sample data
  - The specific (x_i, y_i) pairs in the out-of-sample set

10 different random train/test splits

split in/out of sample	RMSE
1	991.4038
2	962.0368
3	1046.1707
4	969.6299
5	1014.3368
6	984.9417
7	973.5467
8	1044.0364
9	1050.0654
10	947.8253

EMSE across multiple values of K



K-fold cross validation

A more efficient solution is K-fold cross-validation:

1. Randomly divide the data set into K nonoverlapping groups, or folds, of roughly equal size.

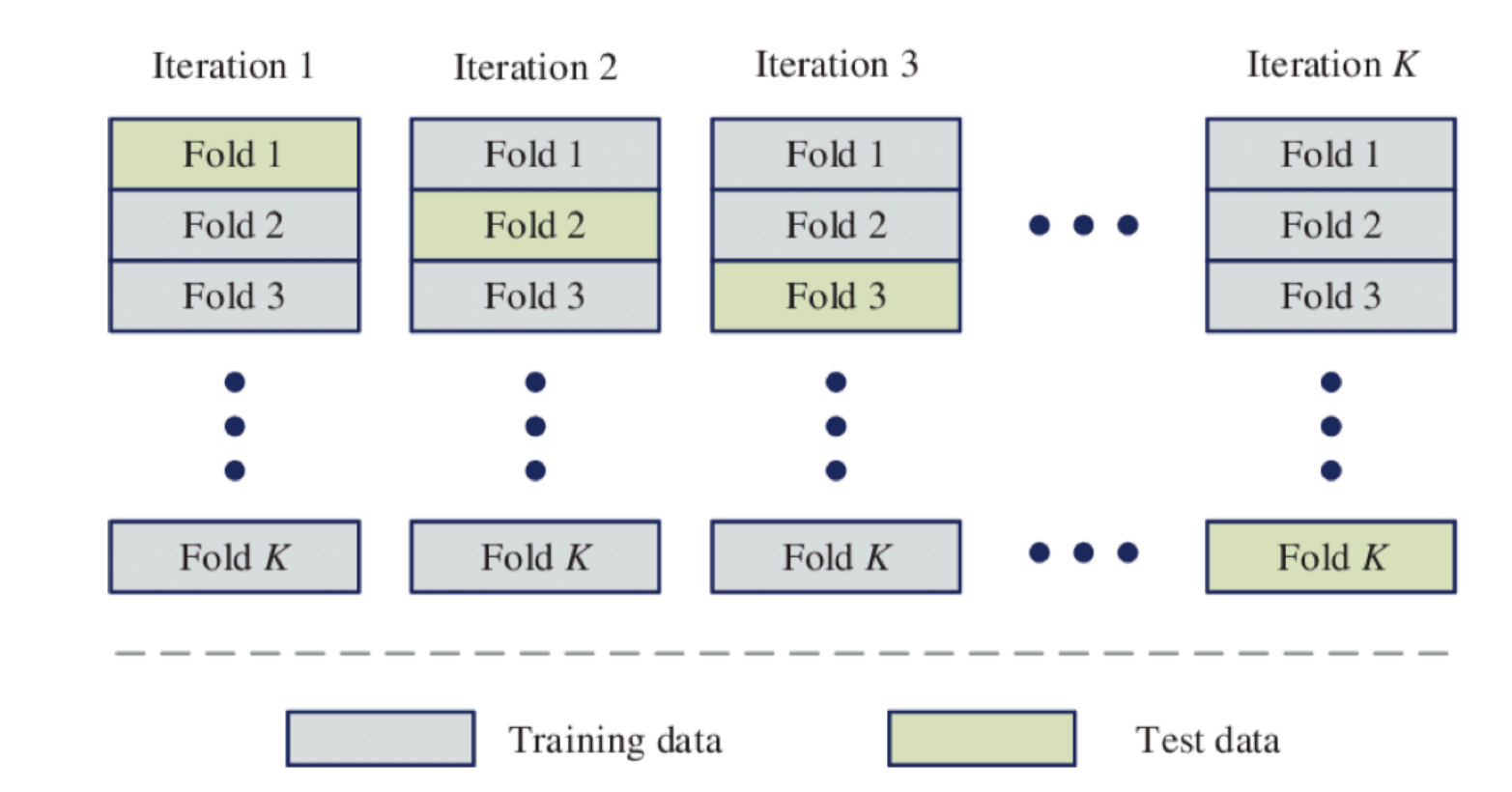
2. For fold $k = 1$ to K:

    - Fit the model using all data points not in fold k.
    - For all points (y_i, x_i) in fold k, predict $\hat{y}_i$ using the fitted model.
    - Calculate $\widehat{\text{RMSE}}_k$, the average error on fold k.

3. Calculate the cross-validated error rate as:

$$\text{CV}_{(K)} = \frac{1}{K} \sum_{k=1}^{K} \widehat{\text{EMSE}}_k$$

K-fold cross validation



- The split of the data into folds is still random, but in a way that minimizes the overlap between each test set.

K-fold cross validation in practice

- Typical values of K are 5 or 10 in practice.
- All candidate models should be fit on the same set of folds.
- That is, do not create a different split to evaluate different models.
- If $K = N$, i.e. the size of the data set, the resulting procedure is called "leave-one-out" cross validation (LOOCV).
- Generally k-fold CV with $K = 5$ or $K = 10$ is preferable to LOOCV.
- Then both training and holdout samples are of reasonable size, achieving a balance on both bias and variance.
- LOOCV tends to have a higher variance. This is because the N folds are highly correlated-any two folds always contain almost the same data points.

K-fold cross validation

There are two typical ways to select a model using cross validation:

1. The min rule: choose the model with the best cross-validated error.
2. The 1SE rule: choose the simplest model whose cross-validated error is within one standard error of the minimum.
    - Because the more complex model has a lower cross-validated error, but may also have a higher variance.

For each model, we estimate the standard error of that model's cross-validated EMSE as:

$$\text{S.E} \approx \frac{\text{sd}\left(\widehat{\text{EMSE}}_1, \widehat{\text{EMSE}}_2, \ldots, \widehat{\text{EMSE}}_K\right)}{\sqrt{K}}$$

Wrap up

- The basic idea of Supervised Learning is to make prediction based on the training data.
- The key is to find a good model that can make accurate predictions on the testing data.
- Because we need to split the data into training and testing sets, the data should be large enough.
- Don't forget to use cross-validation to estimate the performance of the model.