THE UNIVERSITY OF TEXAS AT AUSTIN

Introduction to Predictive Models

Book Chapters 1, 2 and 5.

Carlos M. Carvalho
The University of Texas McCombs School of Business

1. Introduction
2. Measuring Accuracy
3. Out-of Sample Predictions
4. Bias-Variance Trade-Off
5. Classification
6. Cross-Validation
7. k-Nearest Neighbors (kNN)

1. Introduction to Predictive Models

Simply put, the goal is to predict a
target variable Y with input variables X!

In Data Mining terminology this is know as **supervised learning** (also called *Predictive Analytics*).

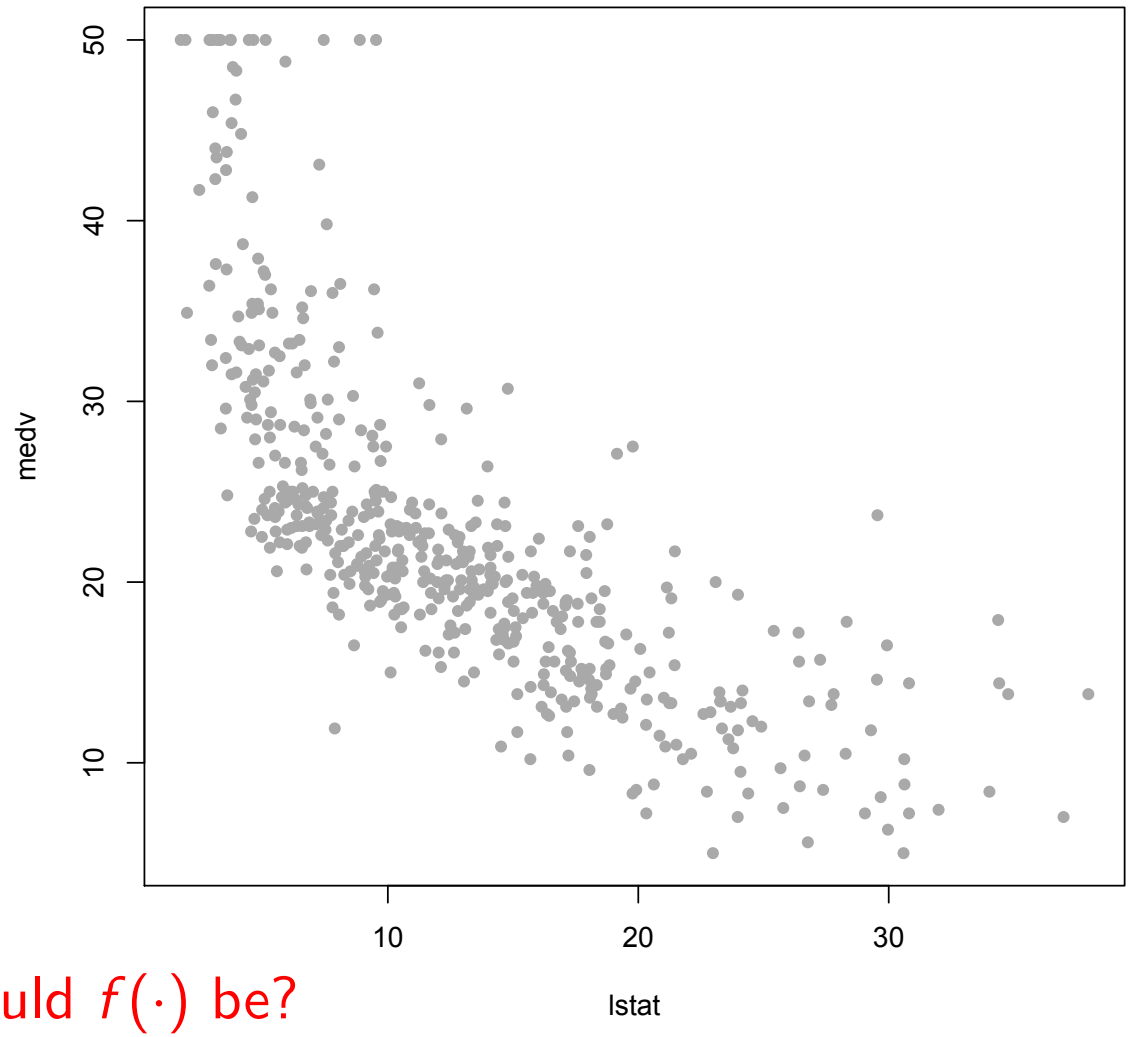
In general, a useful way to think about it is that Y and X are related in the following way:

$$Y_i = f(X_i) + \epsilon_i$$

The main purpose of this part of the course is to *learn or estimate* $f(\cdot)$ from data

Example: Boston Housing

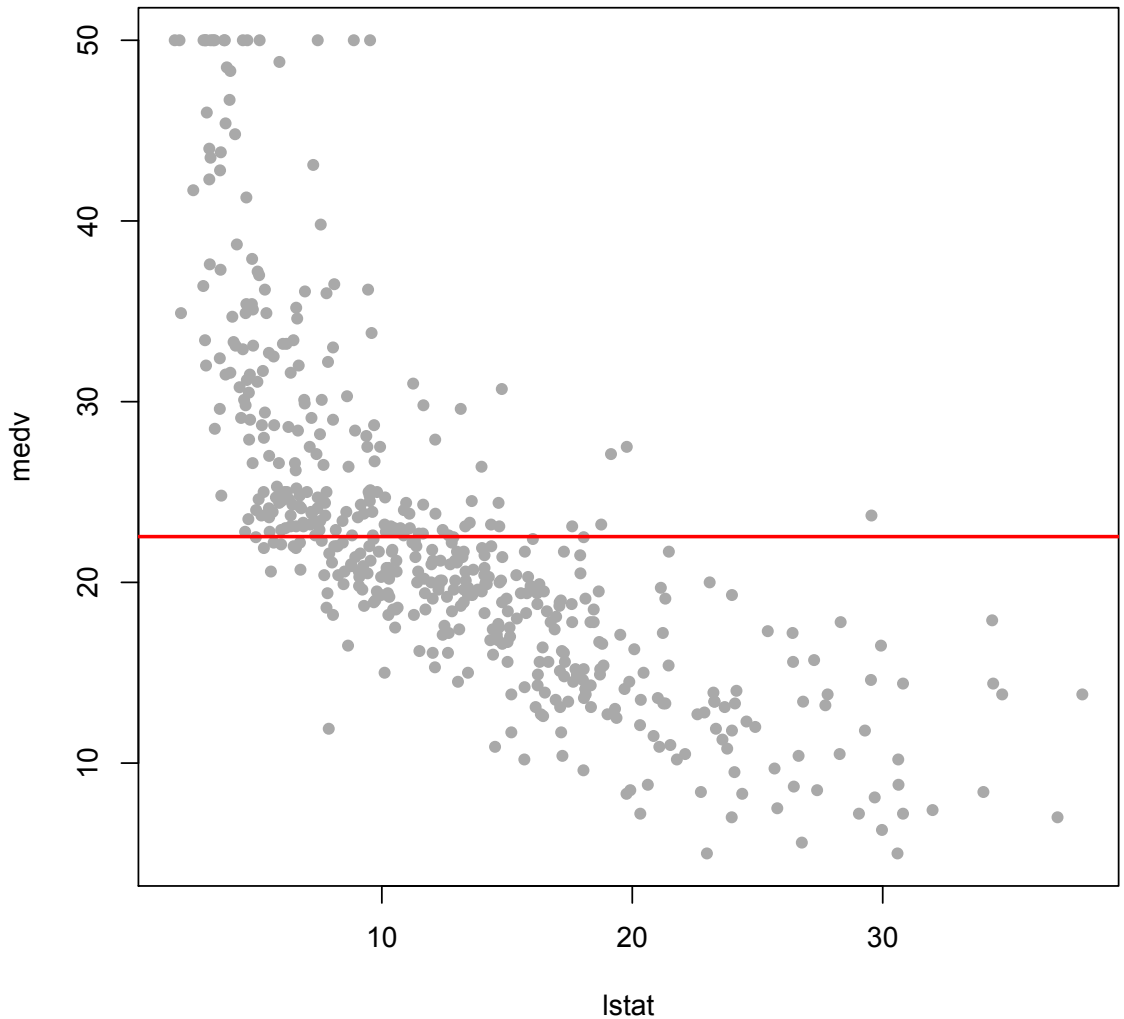
We might be interested in predicting the median house value as a function of some measure of social economic level... here's some data:



What should $f(\cdot)$ be?

Example: Boston Housing

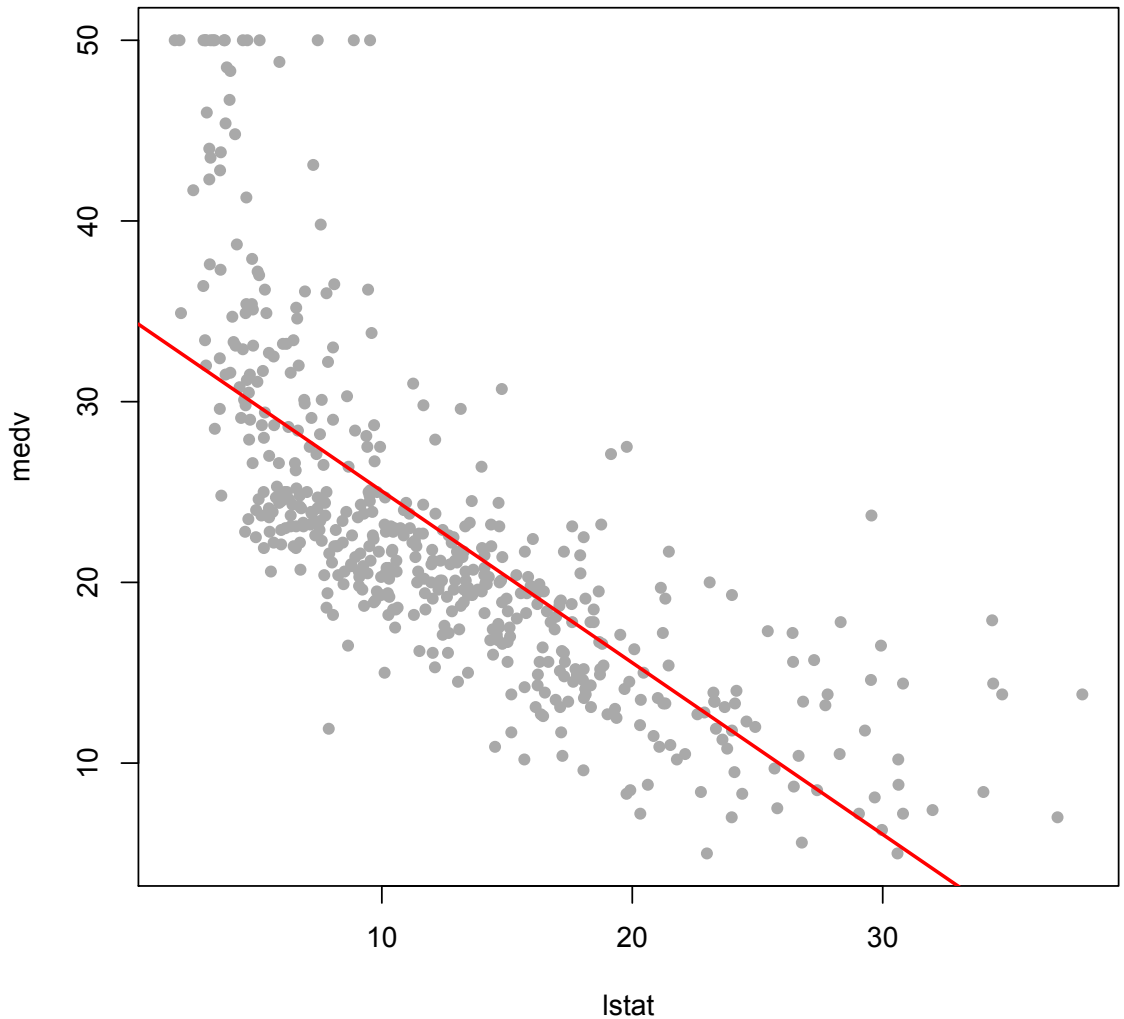
How about this...



If *lstat* = 30 what is the prediction for *medv*?

Example: Boston Housing

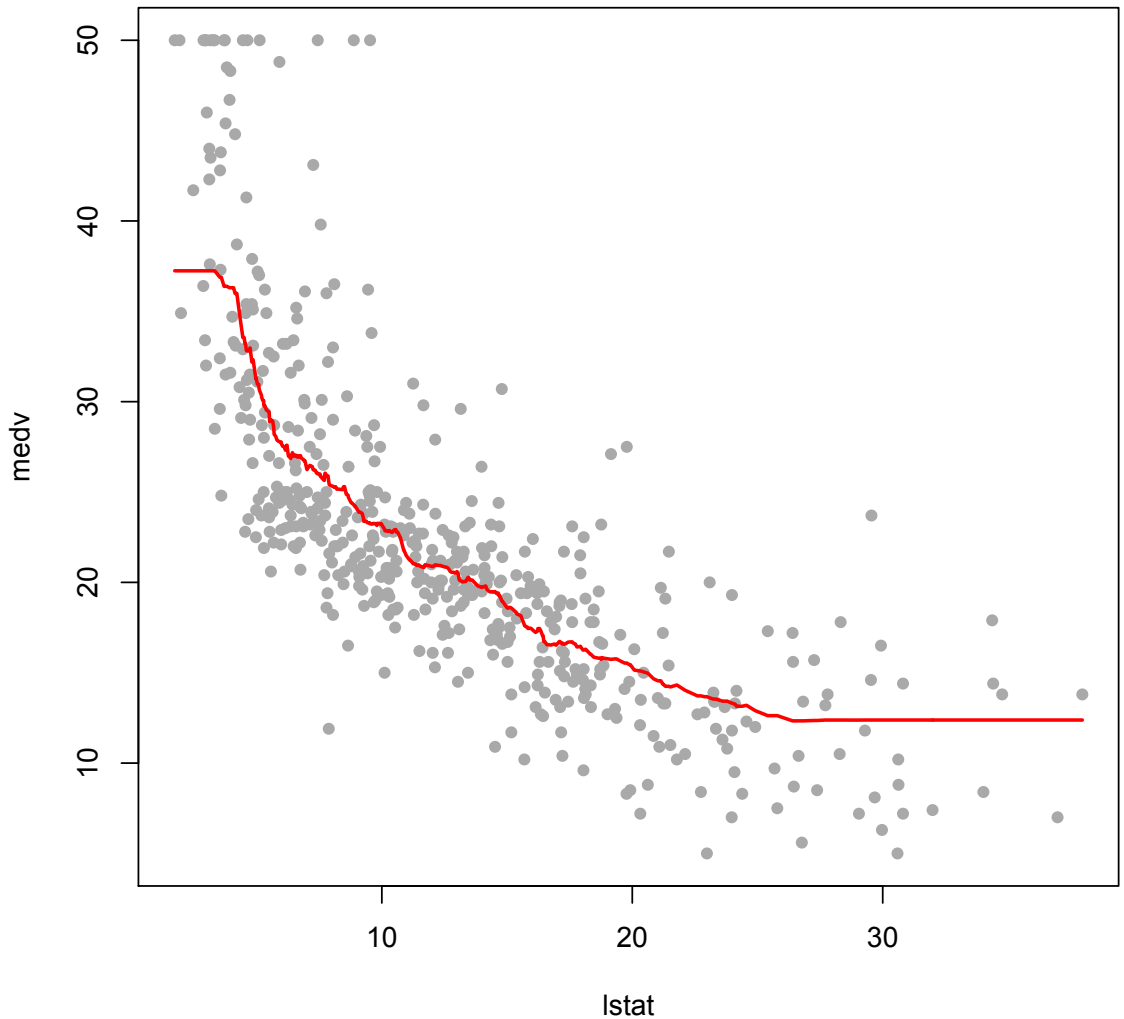
or this...



If *lstat* = 30 what is the prediction for *medv*?

Example: Boston Housing

or even this?



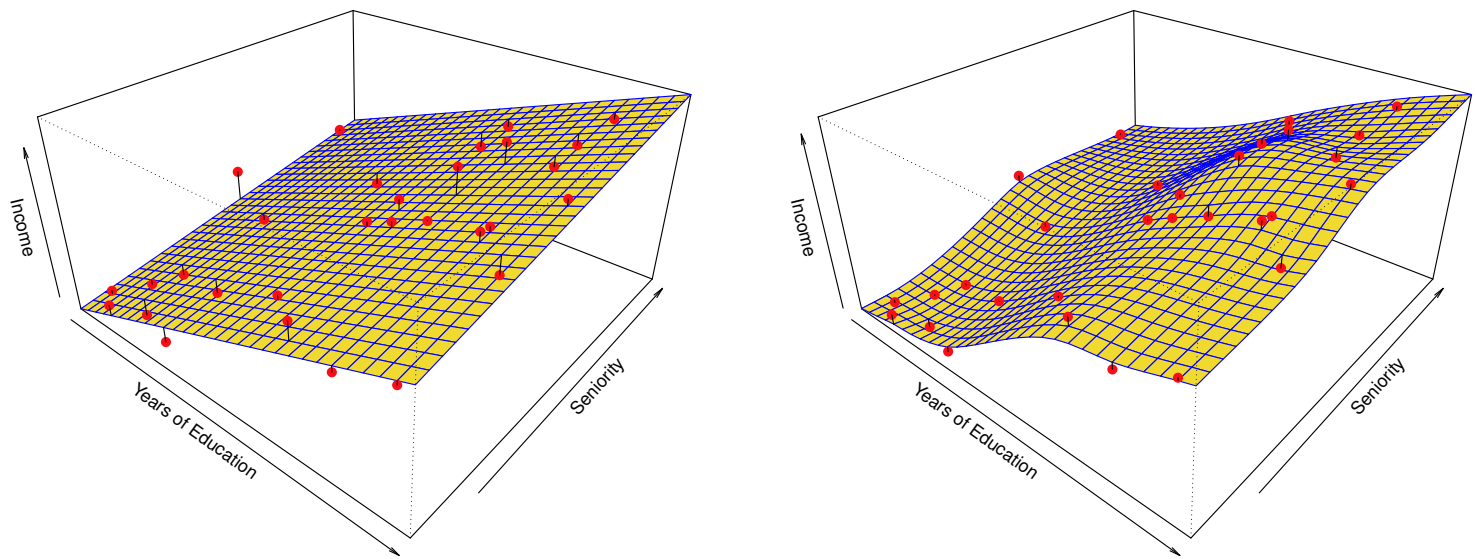
If *lstat* = 30 what is the prediction for *medv*?

How do we estimate $f(\cdot)$?

- Using *training data*:

$$\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$$

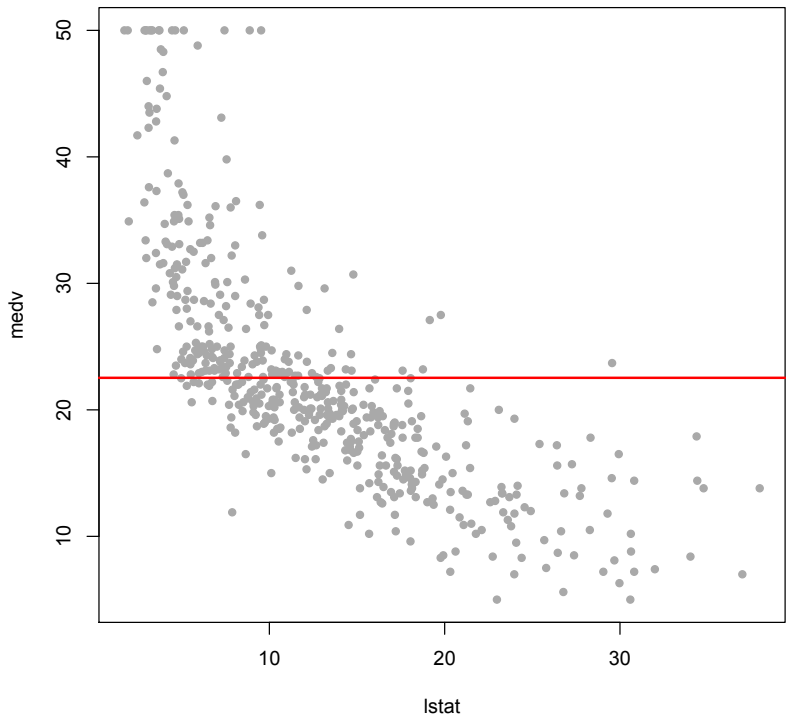
- We use a statistical method to *estimate* the function $f(\cdot)$
- Two general methodological strategies:
  1. parametric models (restricted assumptions about $f(\cdot)$)
  2. non-parametric models (flexibility in defining $f(\cdot)$)



Back to Boston Housing

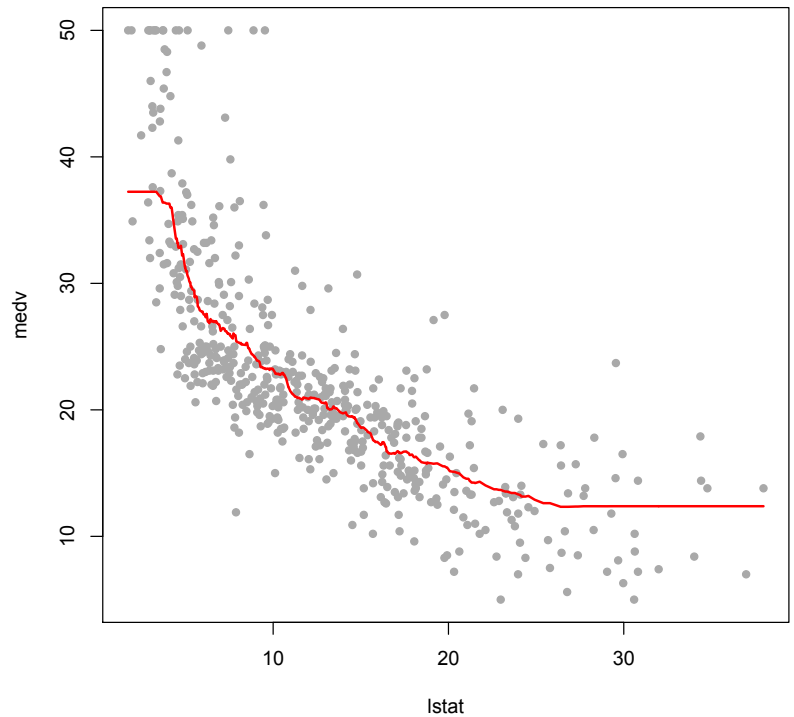
Parametric Model

($Y = \mu + \epsilon$)



Non-Parametric Model

(k-nearest neighbors)



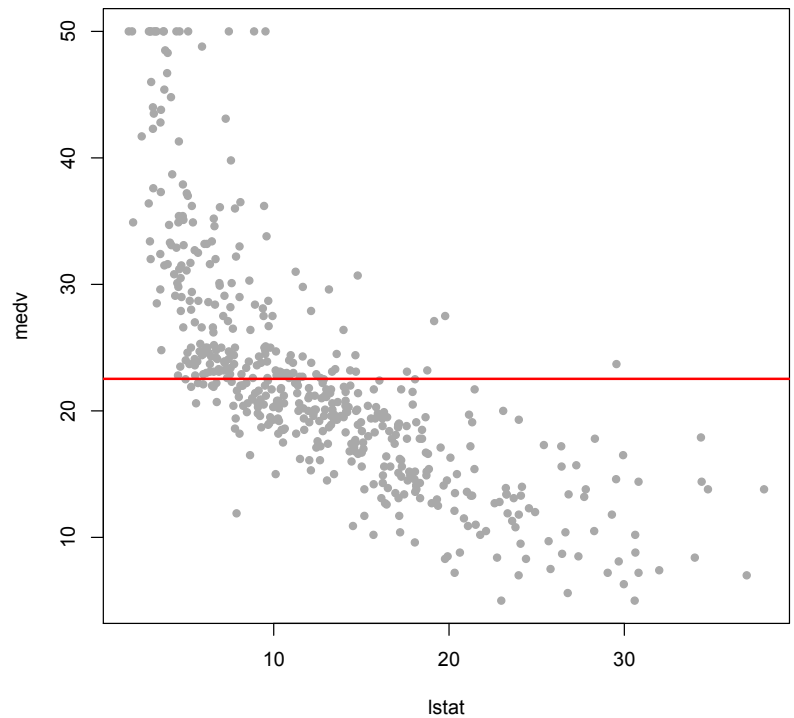
Back to Boston Housing

Simplest parametric model:

$$Y_i = \mu + \epsilon_i$$

Using the training data, we estimate $f(\cdot)$ as

$$\widehat{f(\cdot)} = \hat{\mu} = \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$



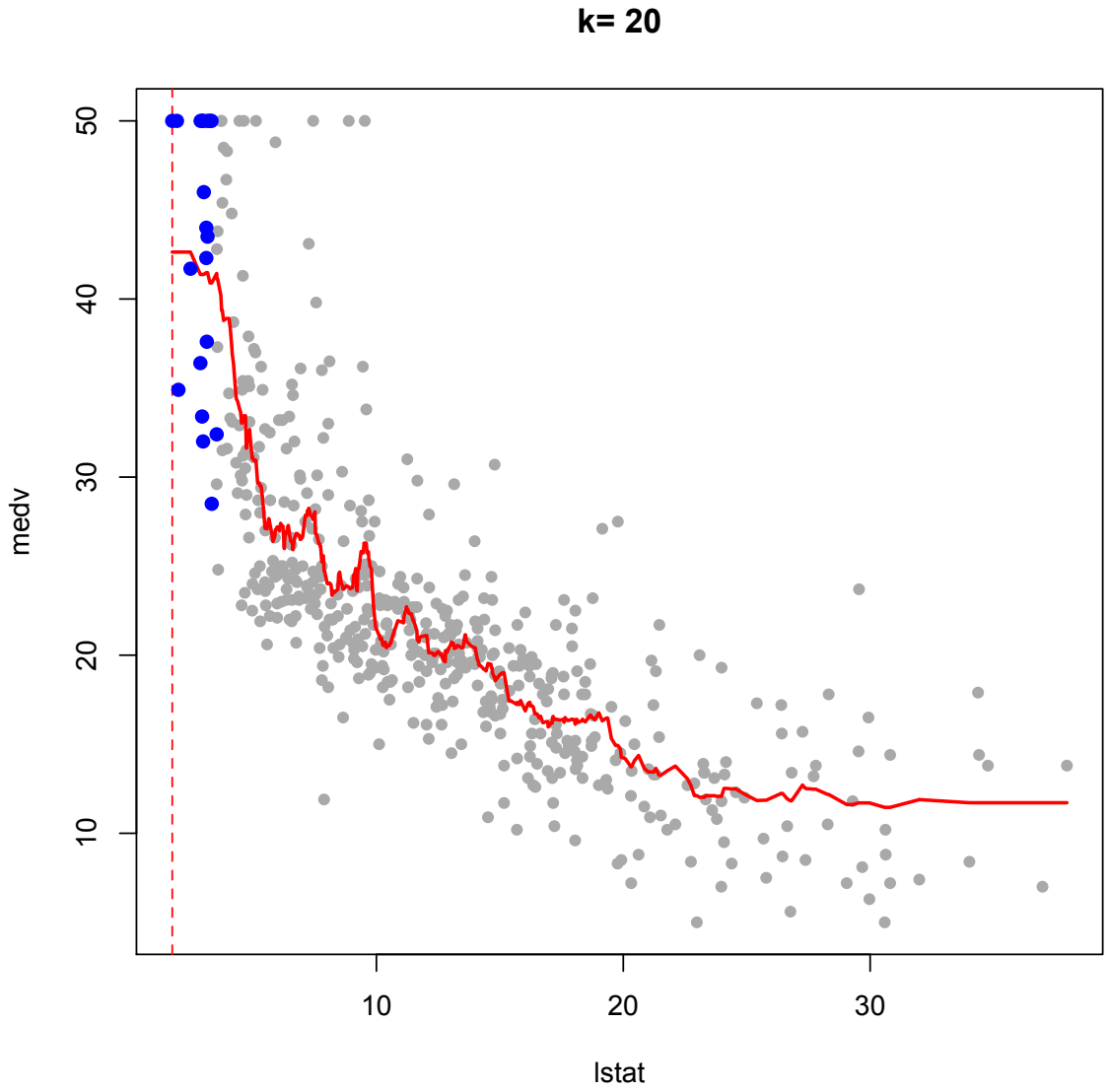
Back to Boston Housing

The above strategy averages all points in the training set... maybe points that are "closer" to the place I am trying to predict should be more relevant...

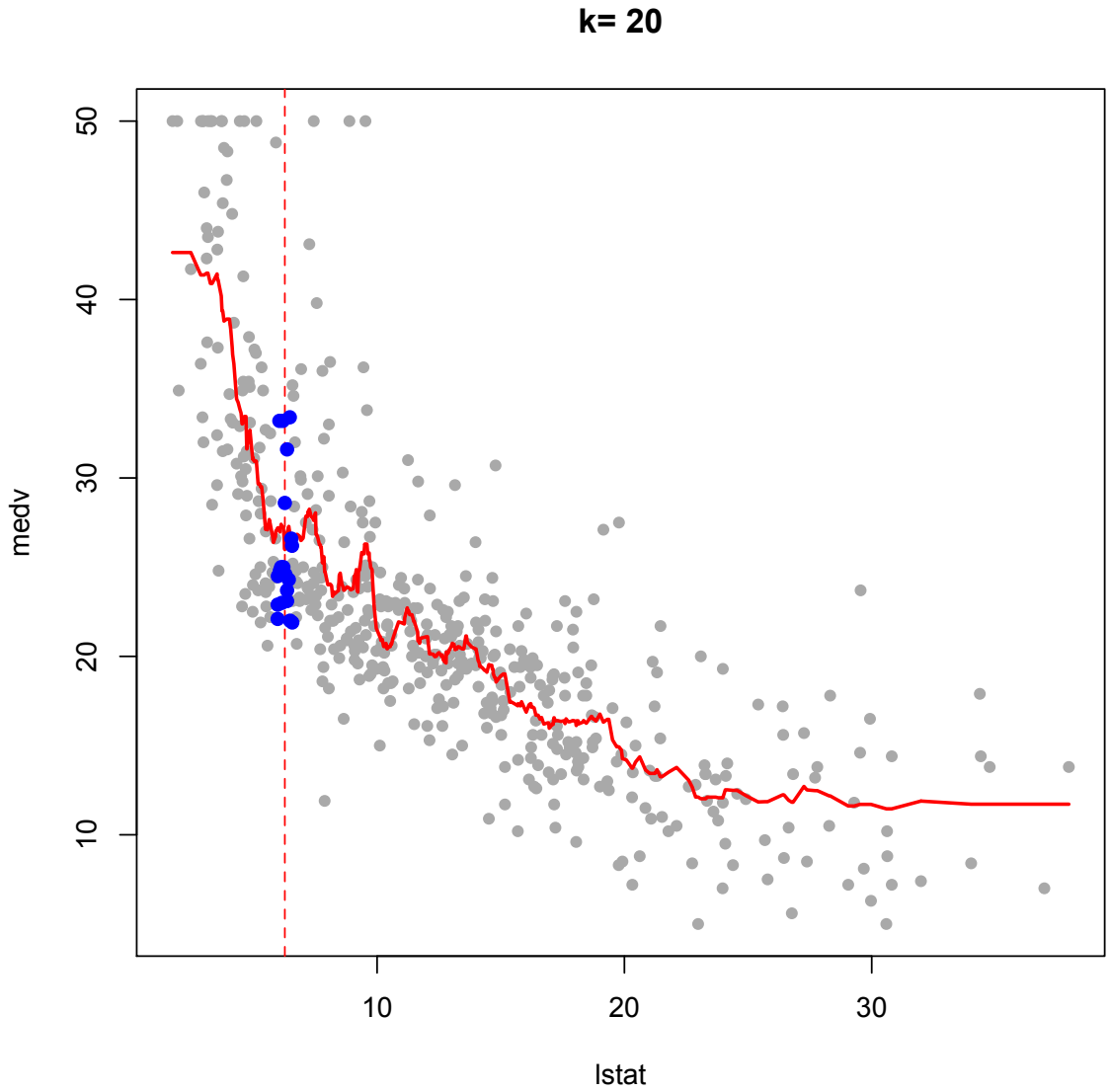
How about averaging the closest 20 neighbors?
What do I mean by closest? We will choose the 20 points that are closest to the X value we are trying to predict.

This is what is called the *k*-nearest neighbors algorithm

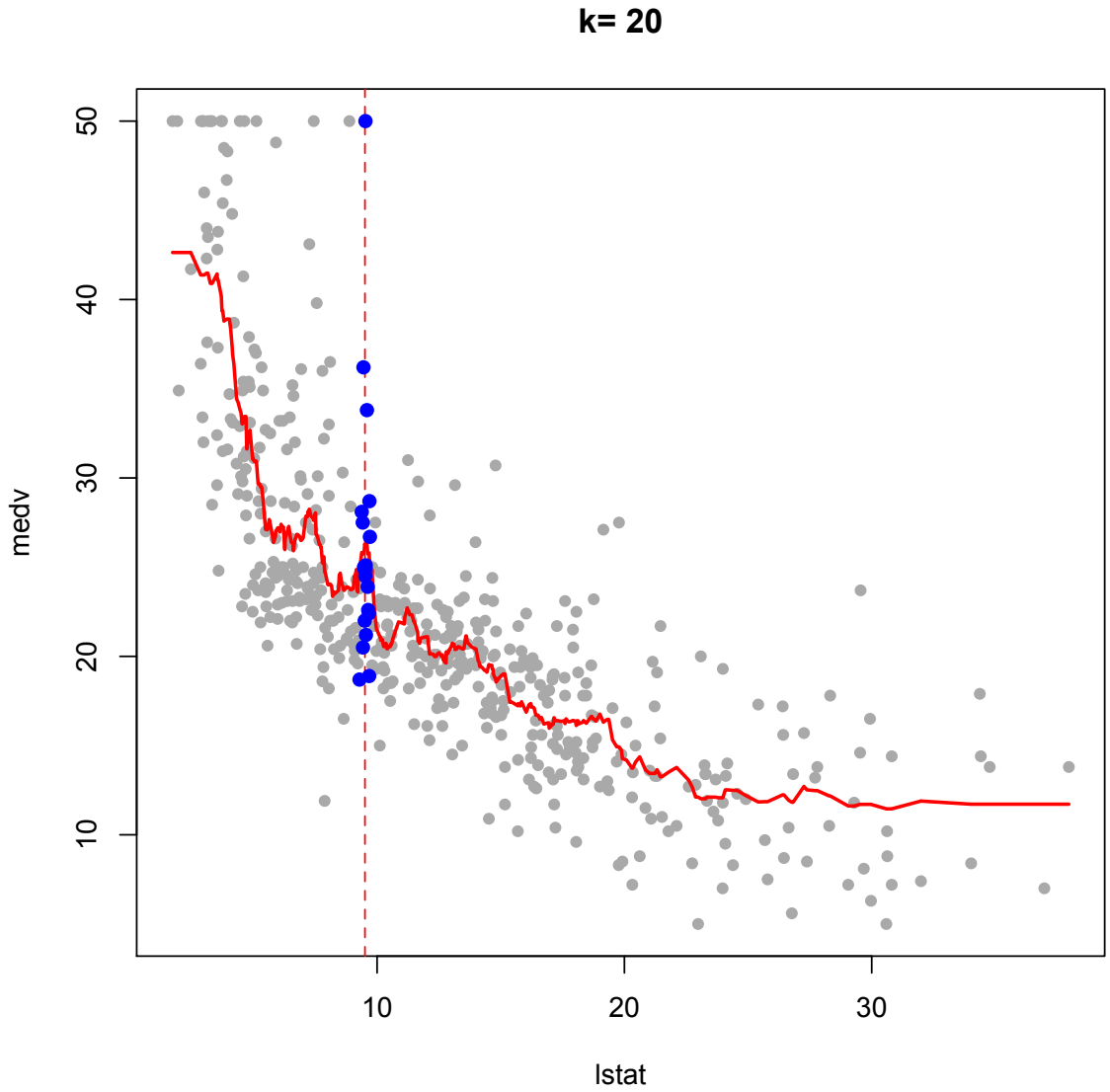
Back to Boston Housing



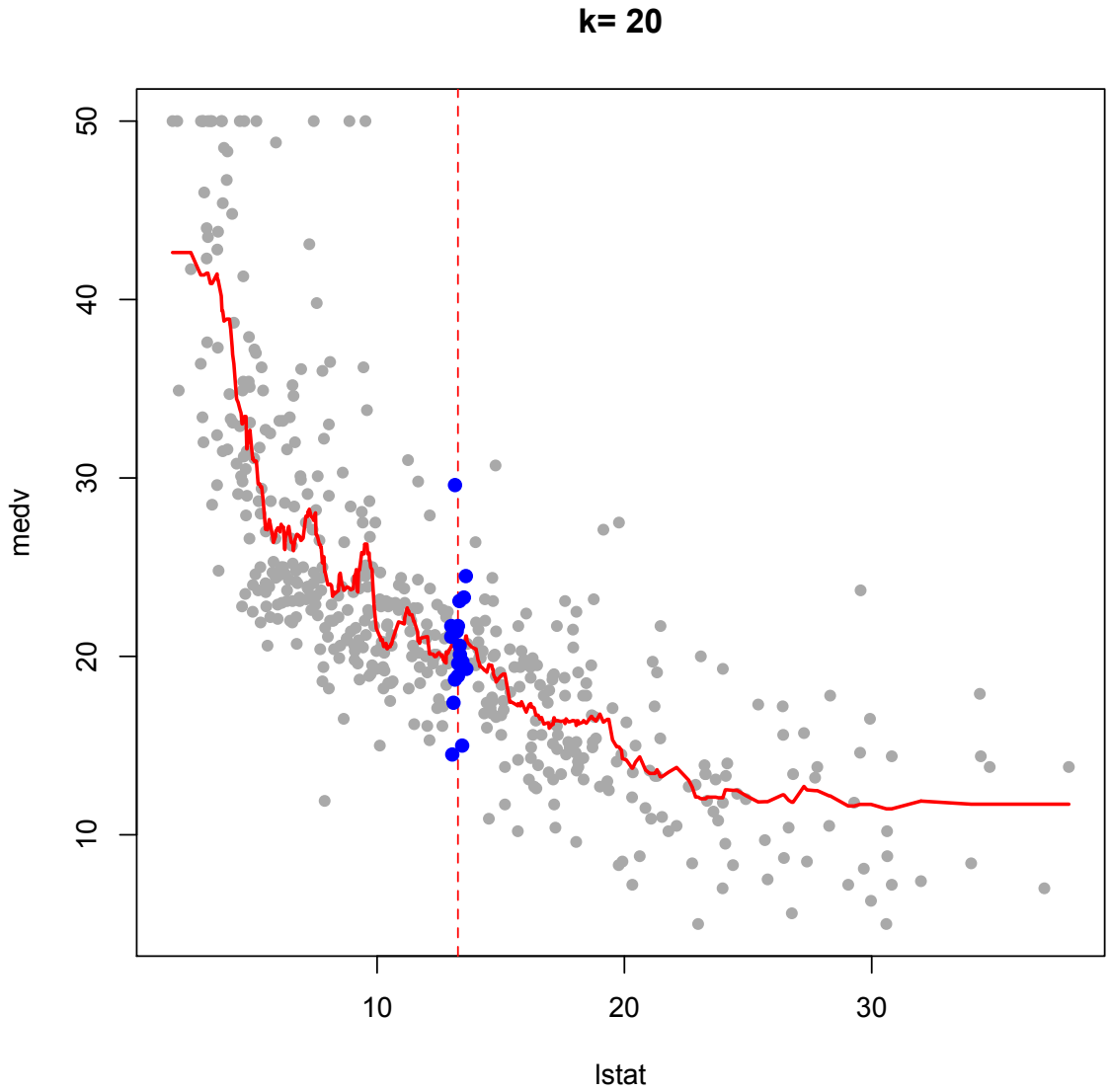
Back to Boston Housing



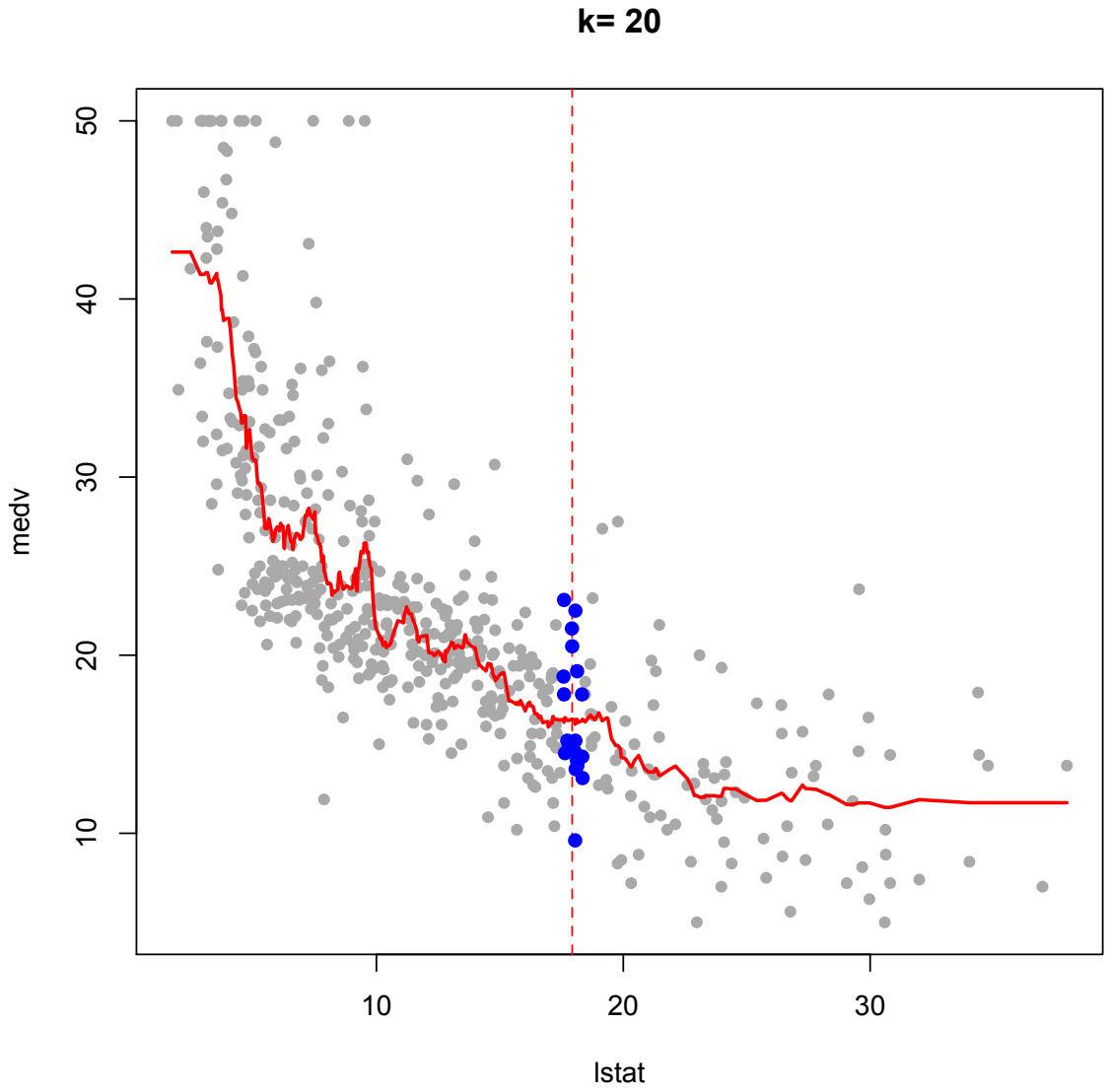
Back to Boston Housing



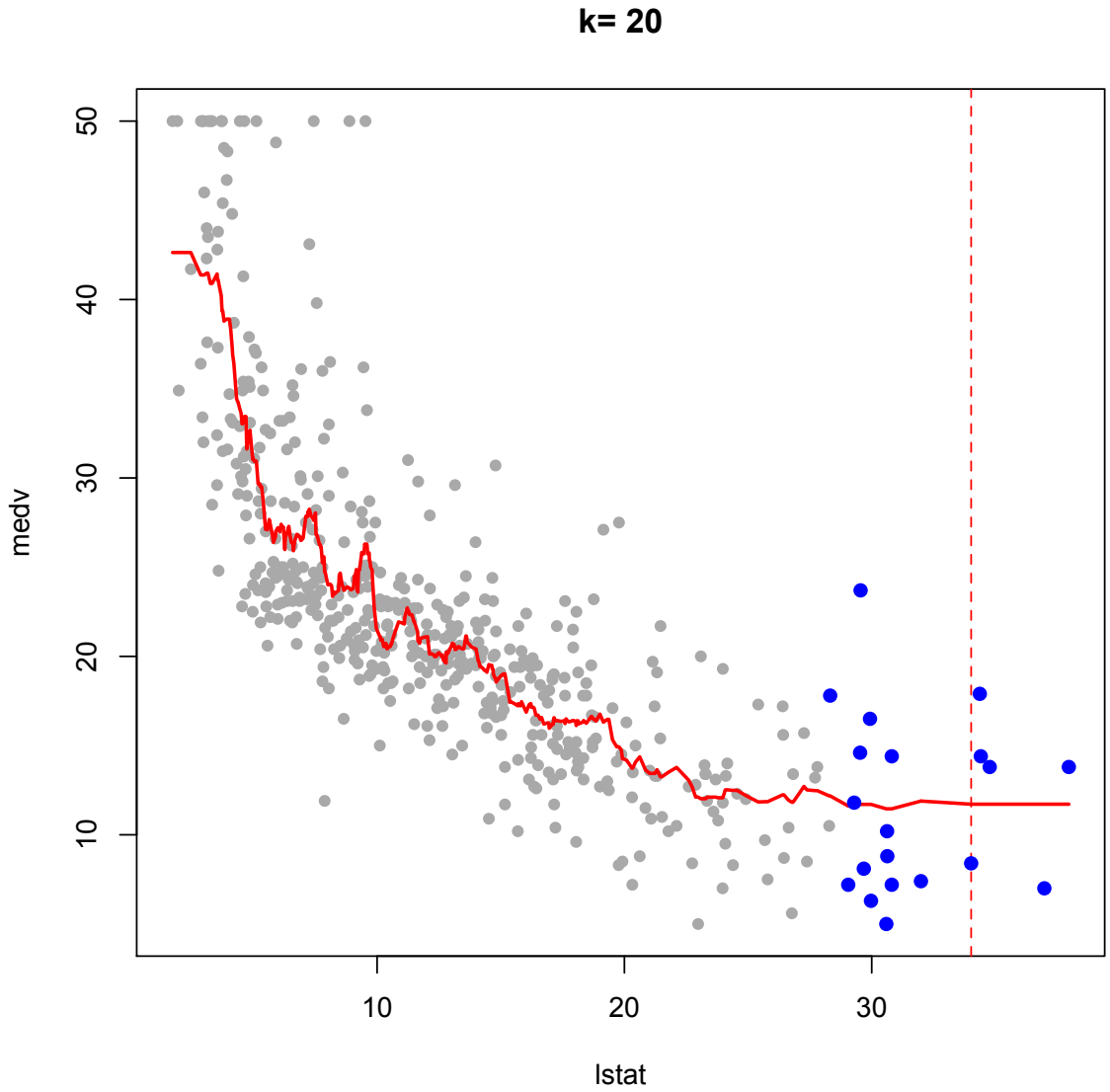
Back to Boston Housing



Back to Boston Housing

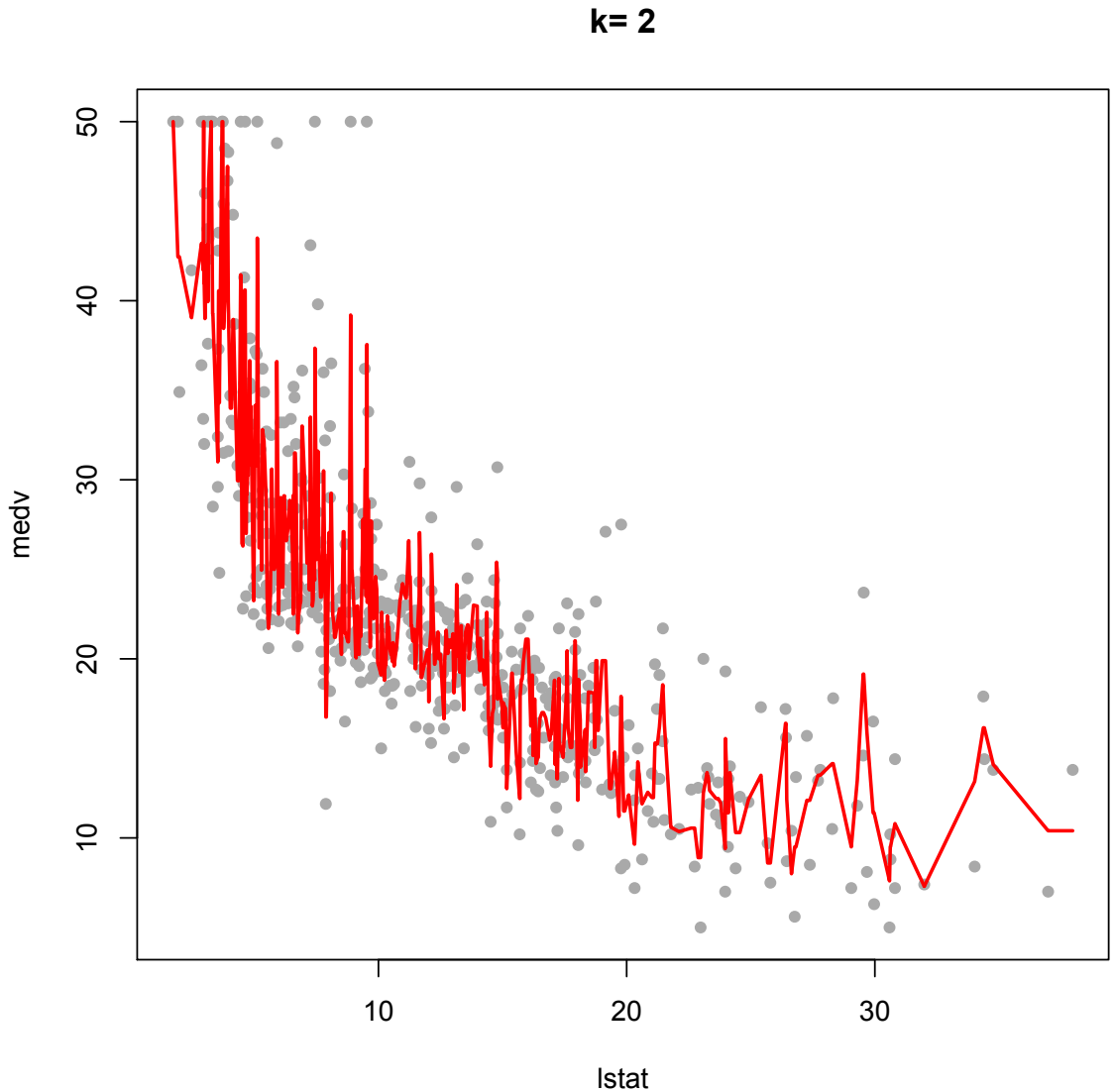


Back to Boston Housing



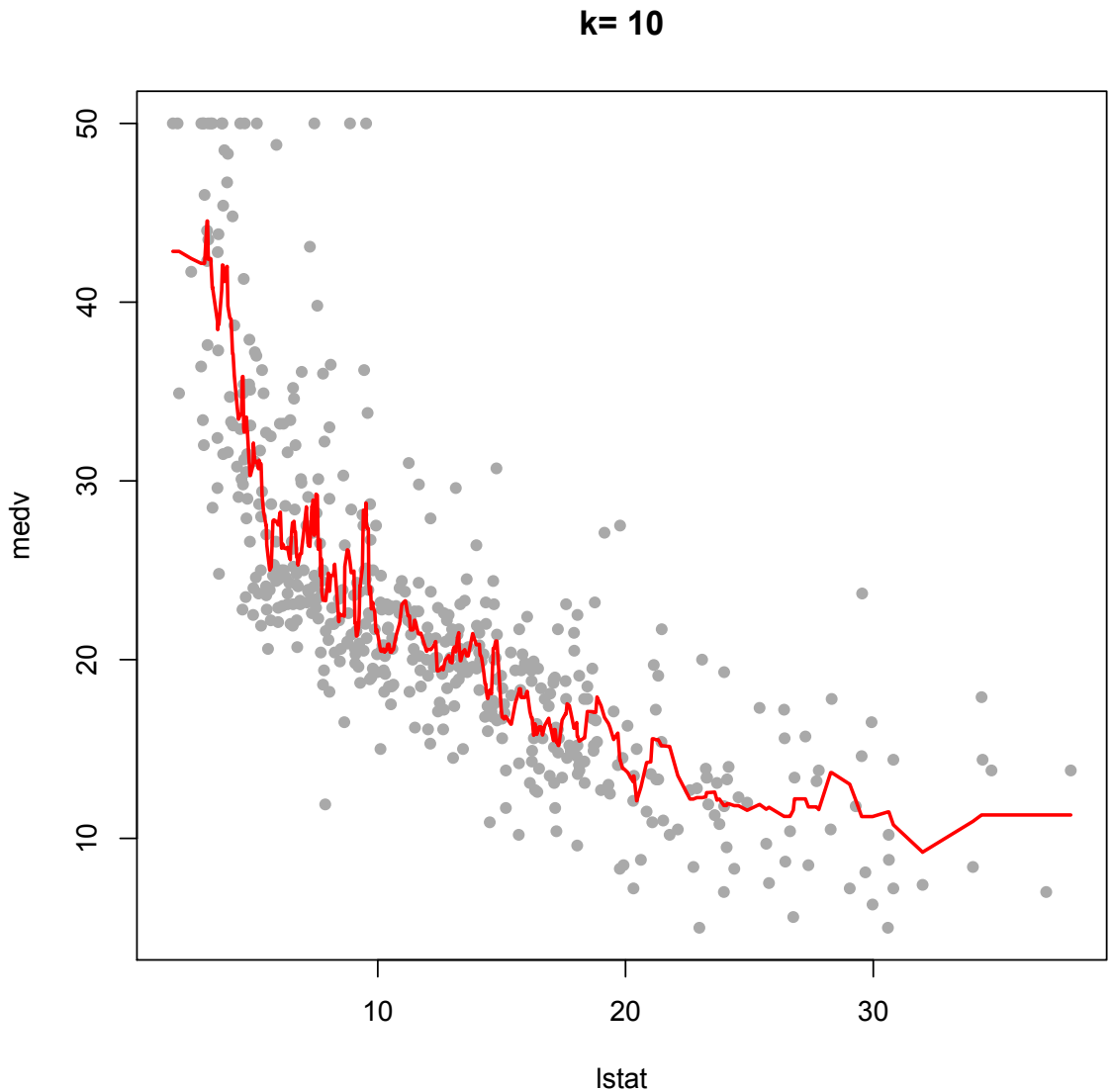
Back to Boston Housing

Okay, that seems sensible but why not use 2 neighbors or 200 neighbors?



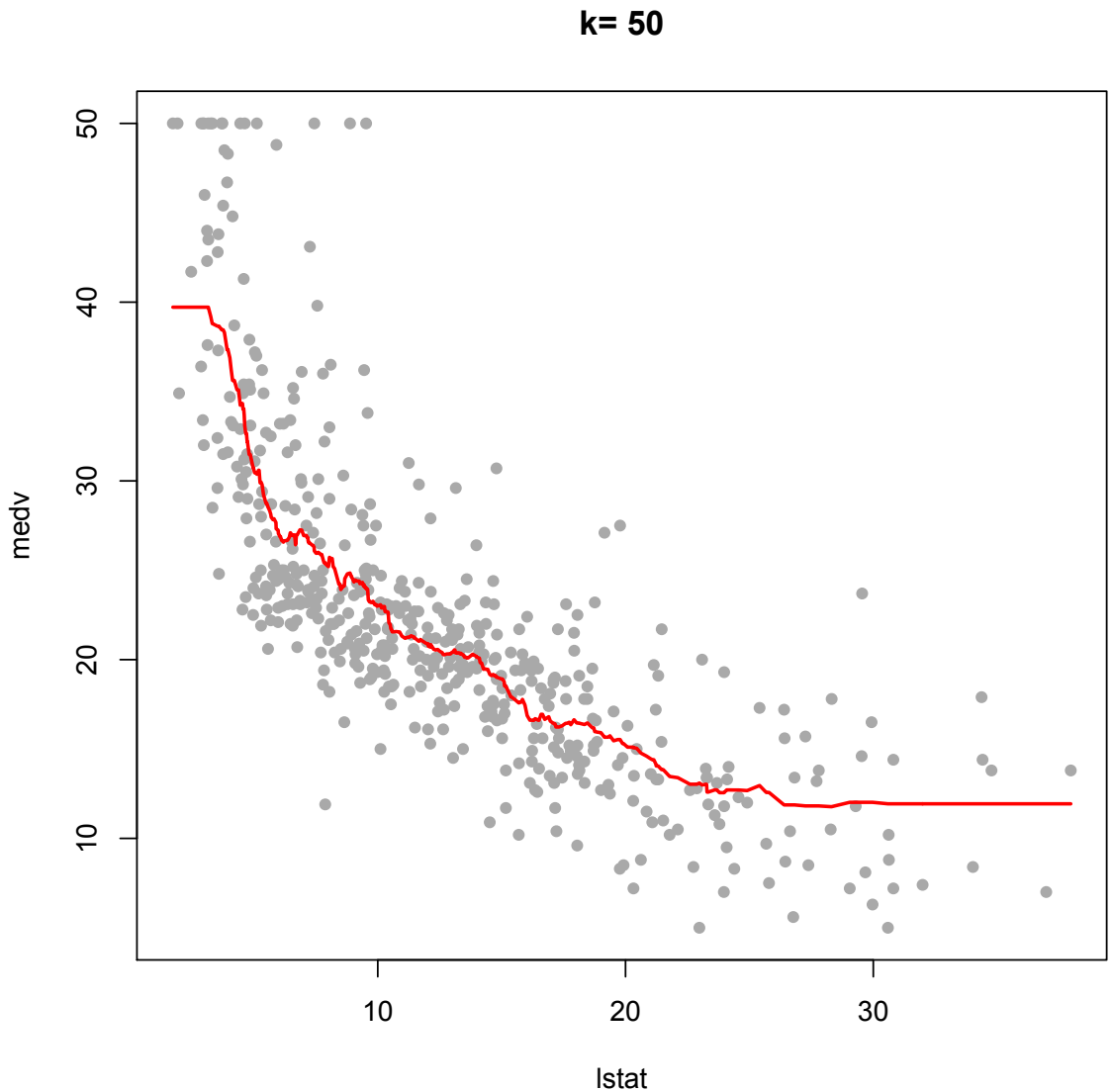
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



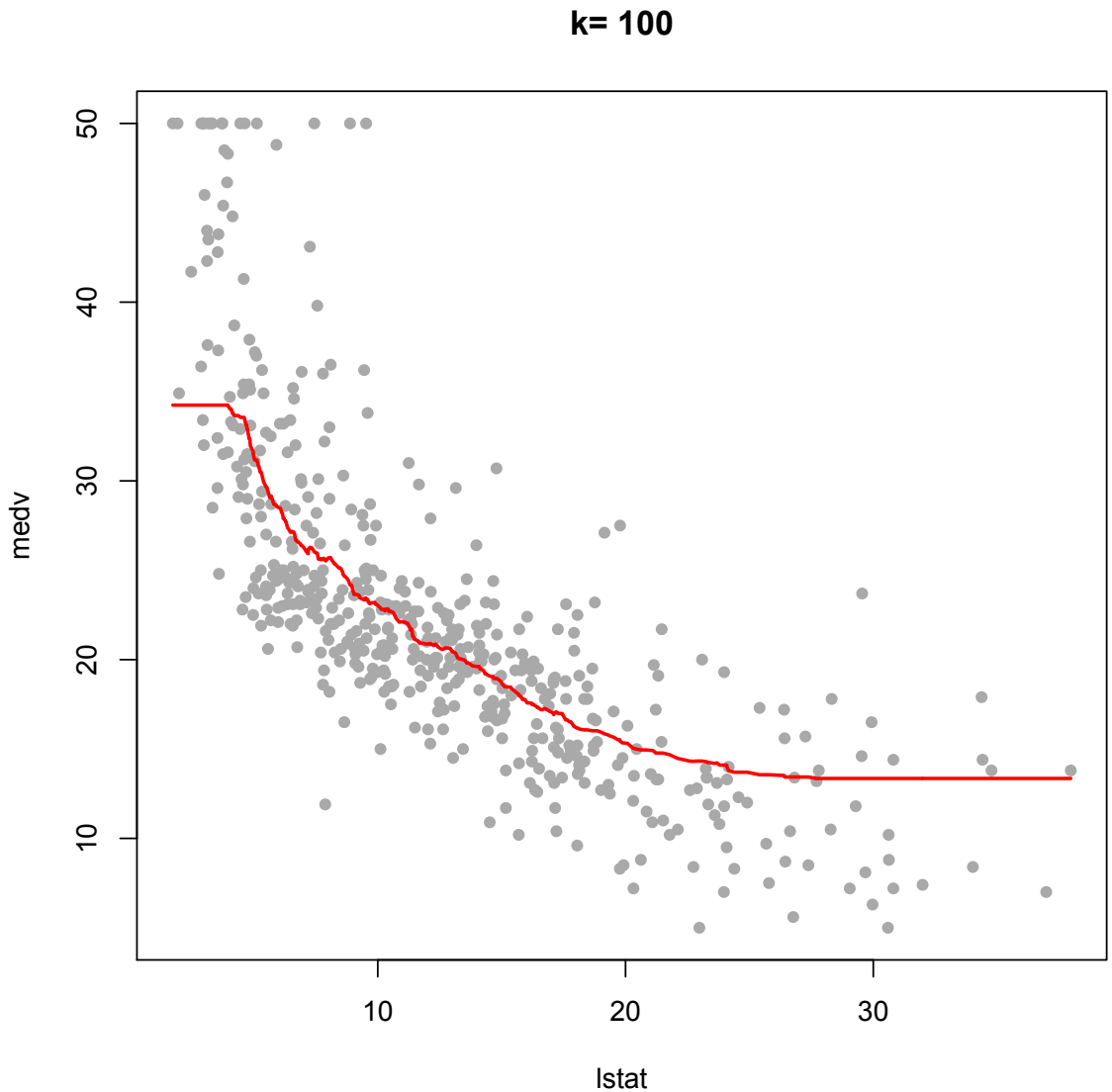
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



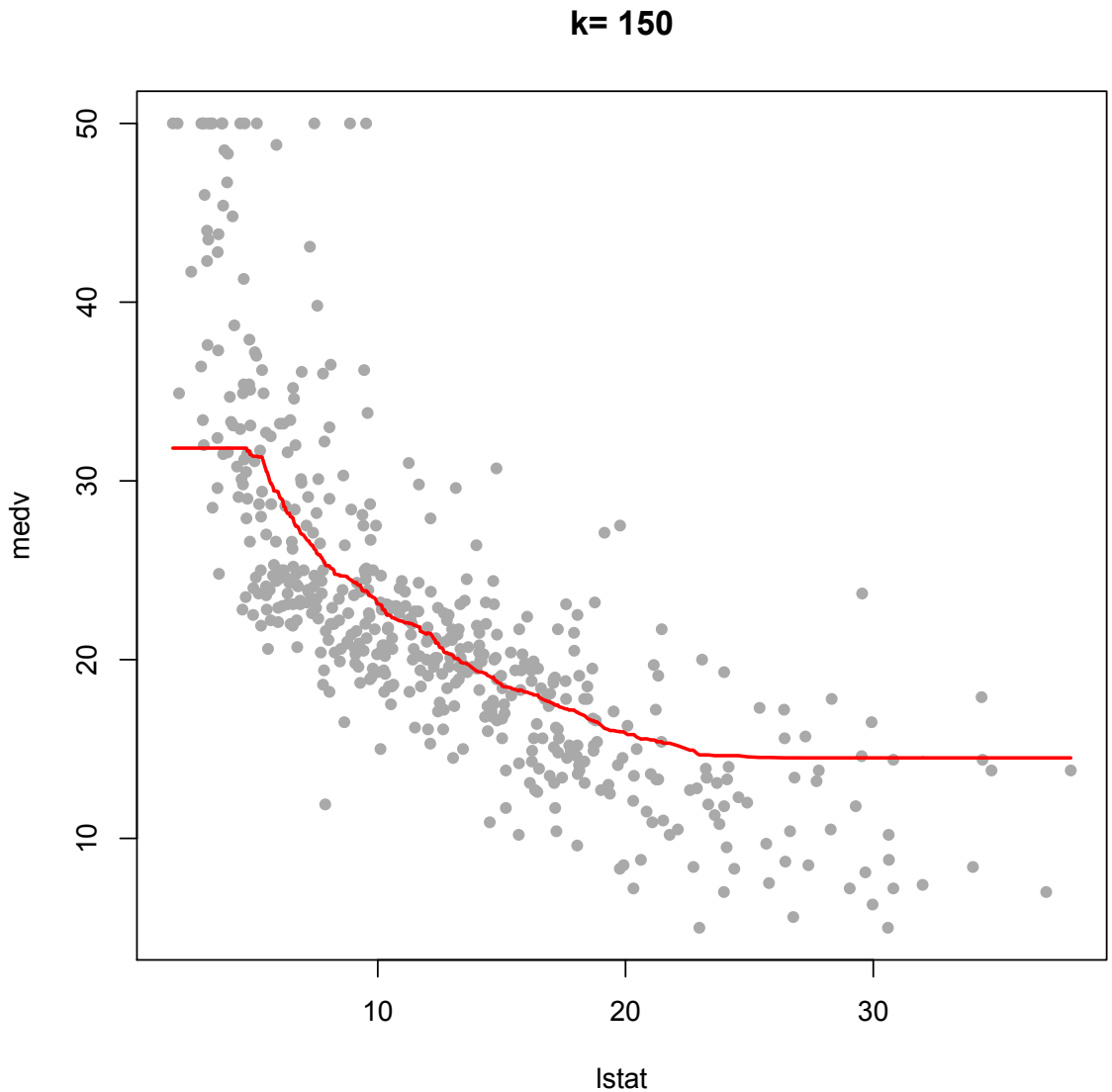
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



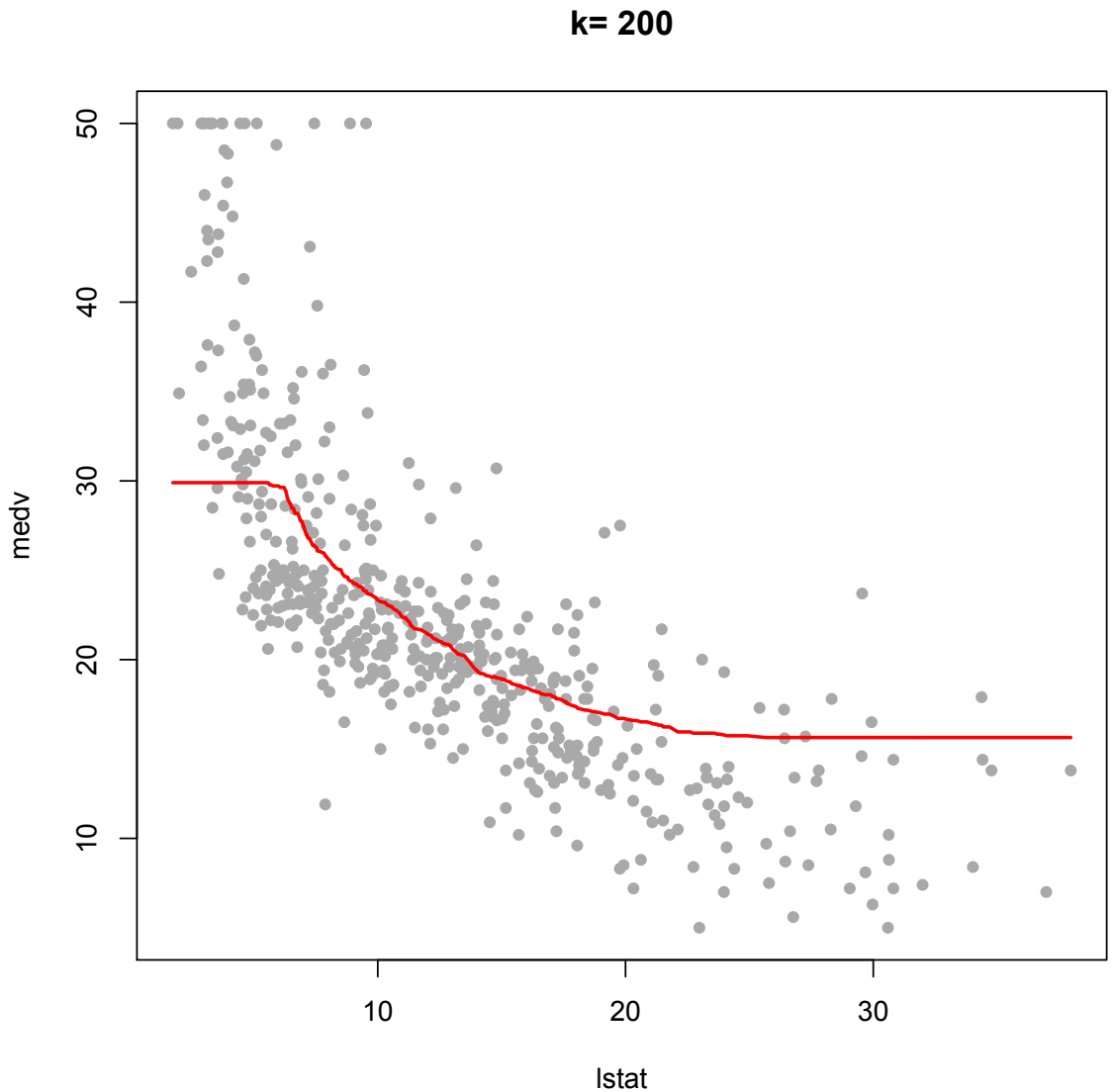
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



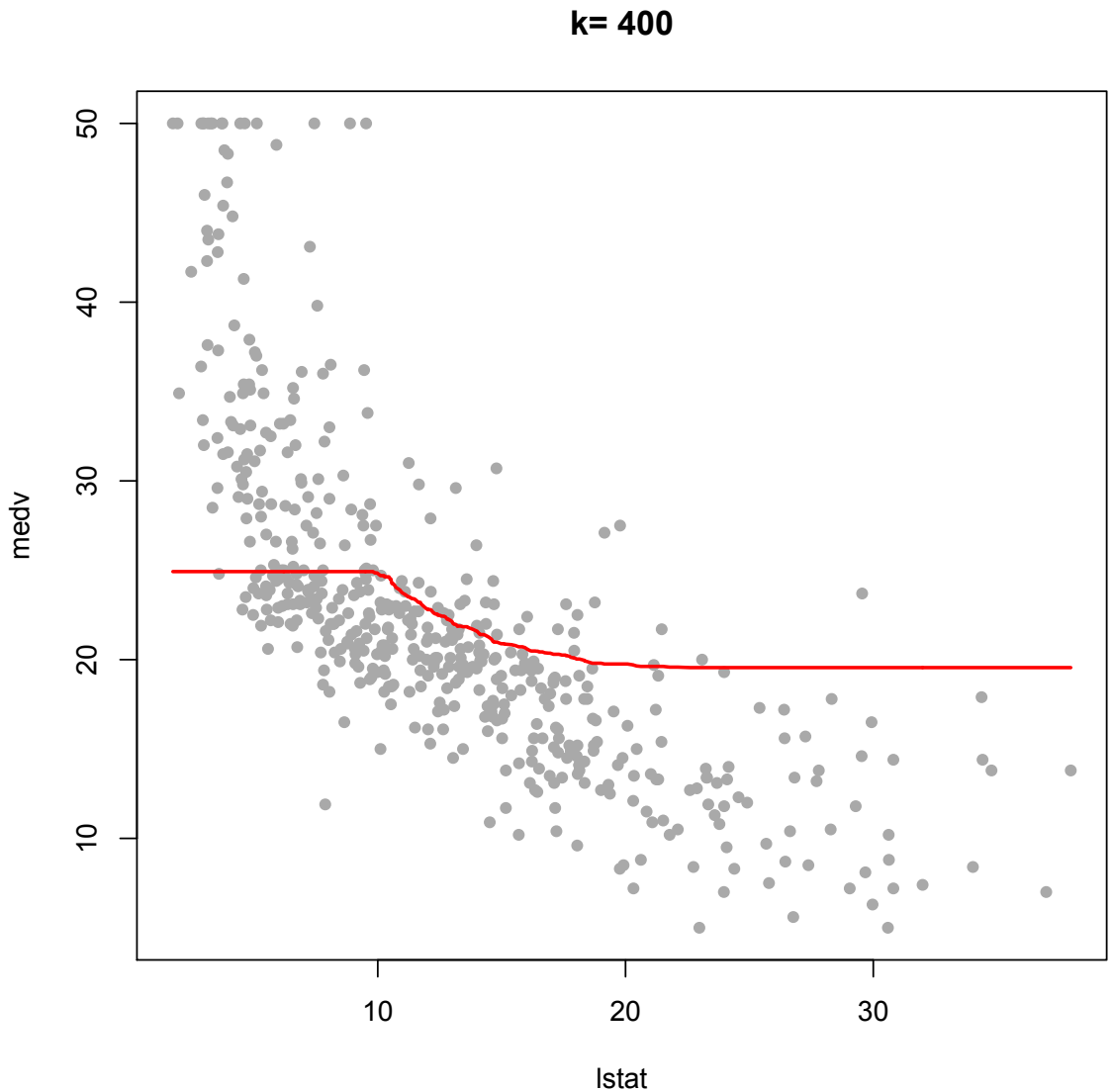
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



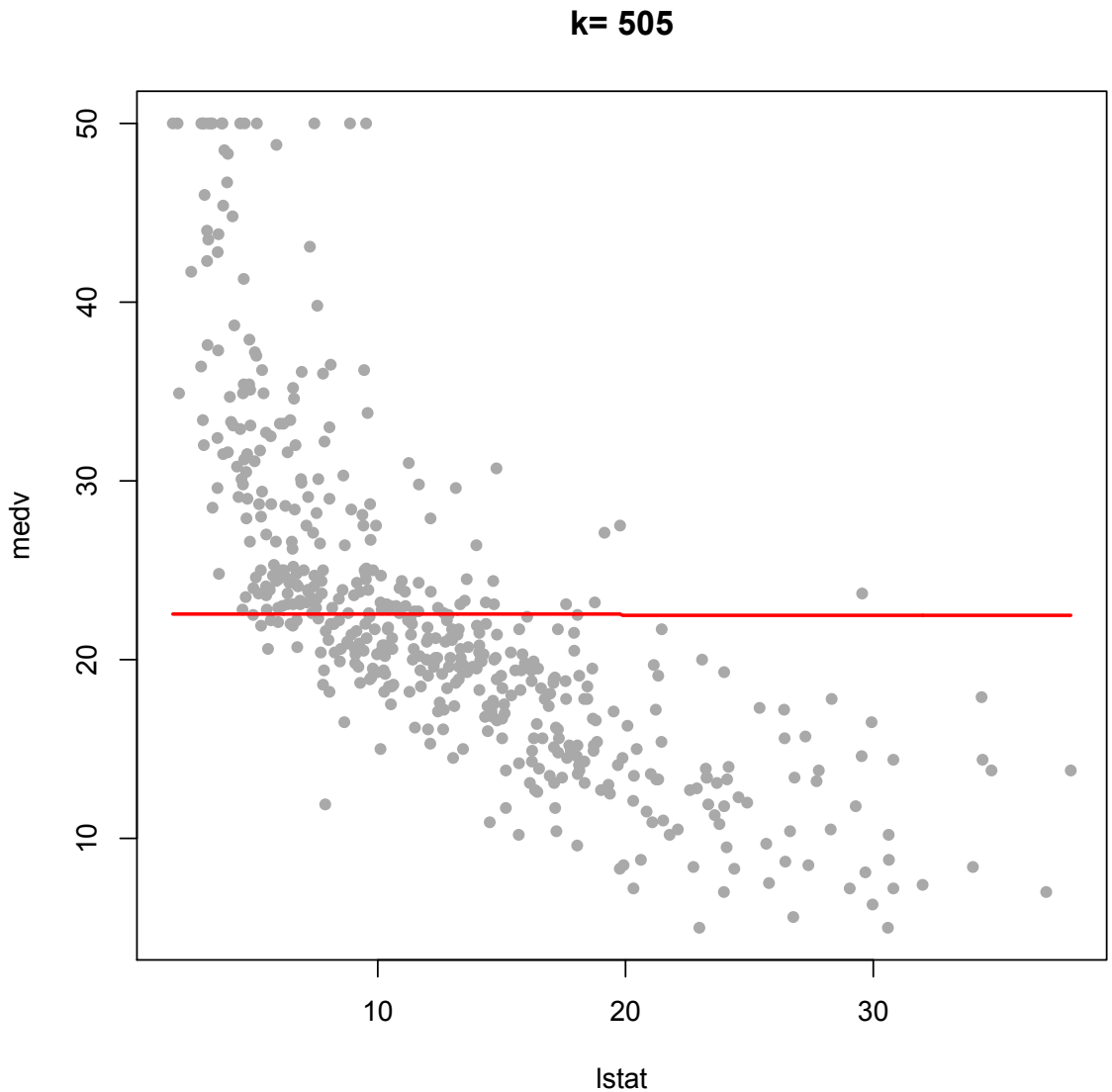
Back to Boston Housing

Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?



Back to Boston Housing

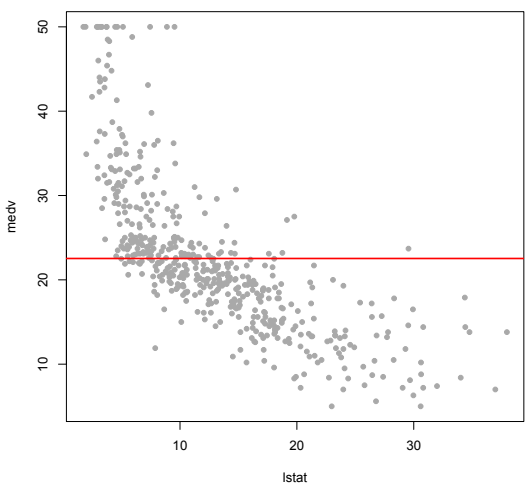
Okay, that seems sensible but why not use 5 neighbors or 200 neighbors?

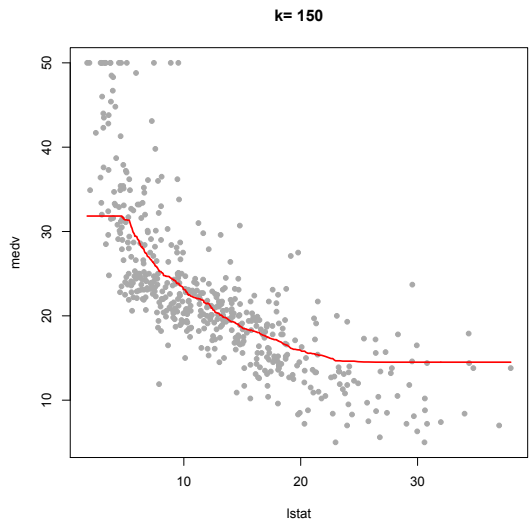


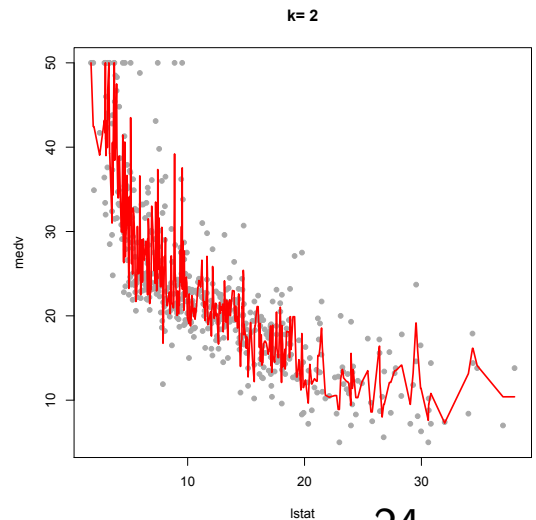
Complexity, Generalization and Interpretation

- As we have seen in the examples above, there are lots of options in estimating $f(X)$.
- Some methods are very flexible some are not... *why would we ever choose a less flexible model?*
  1. Simple, more restrictive methods are usually easier to interpret
  2. More importantly, it is often the case that simpler models are more accurate in making future predictions.

Not too simple, but not too complex!







2. Measuring Accuracy

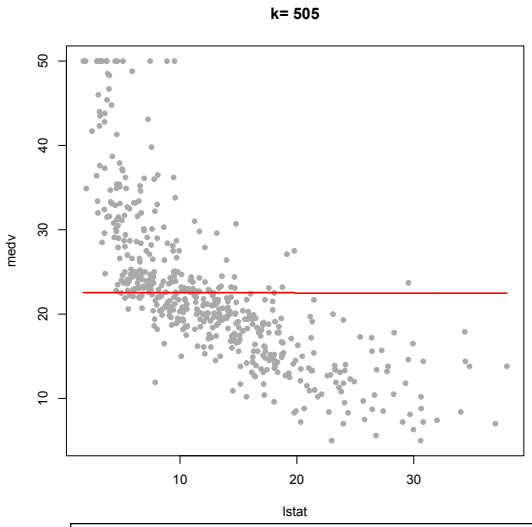
How accurate are each of these models?

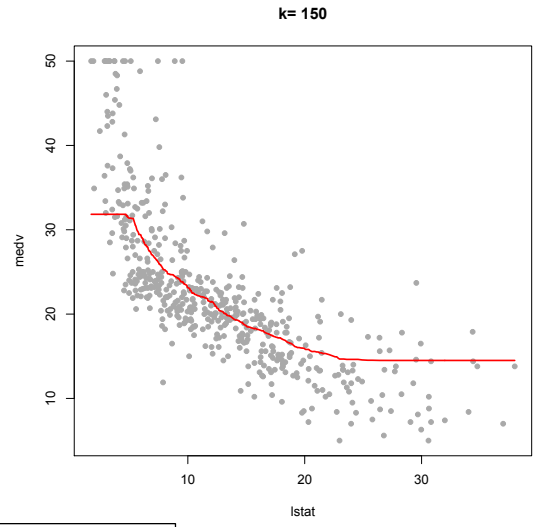
Using the training data a standard measure of accuracy is the *root mean-squared error*

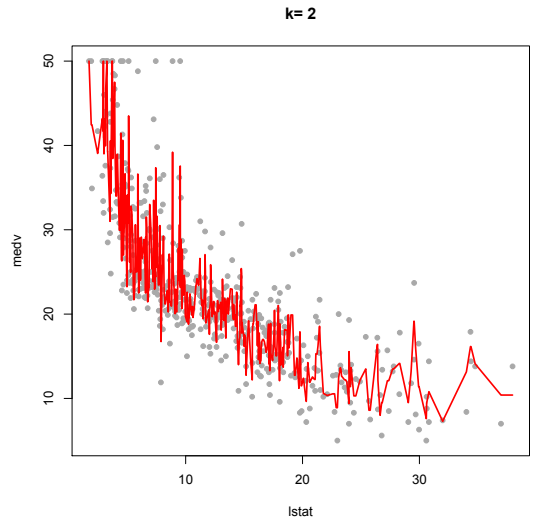
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \widehat{f(X_i)}\right]^2}$$

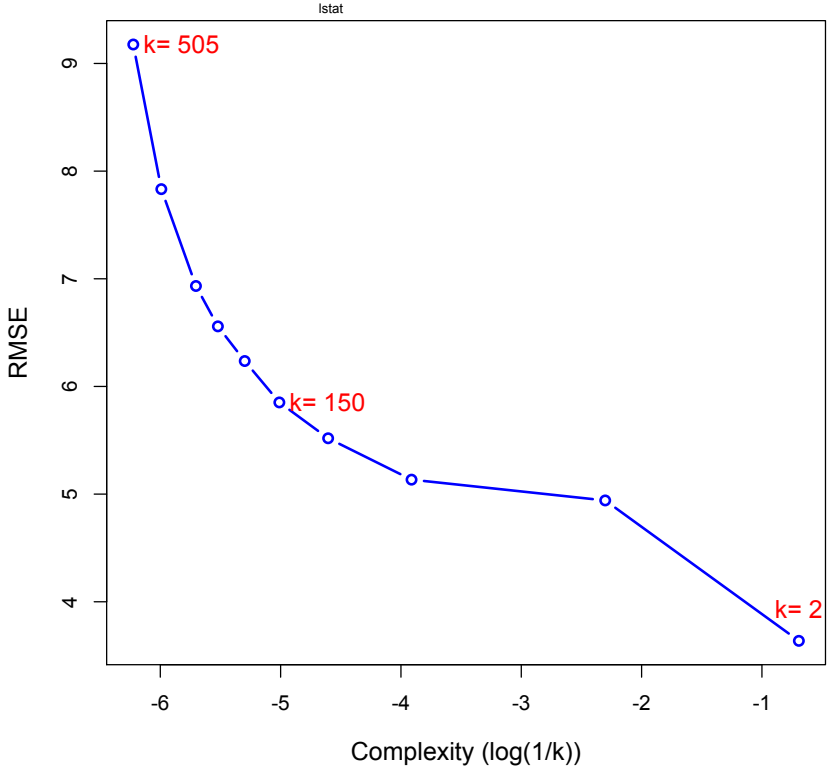
This measure, on average, how large the "mistakes" (errors) made by the model are...

Measuring Accuracy (Boston housing, again)









So, I guess we should just go with the most complex model, i.e., $k = 2$, right?

3. Out-of Sample Predictions

But, do we really care about explaining what we have already seen?

Key Idea: what really matters is our prediction accuracy out-of-sample!!!

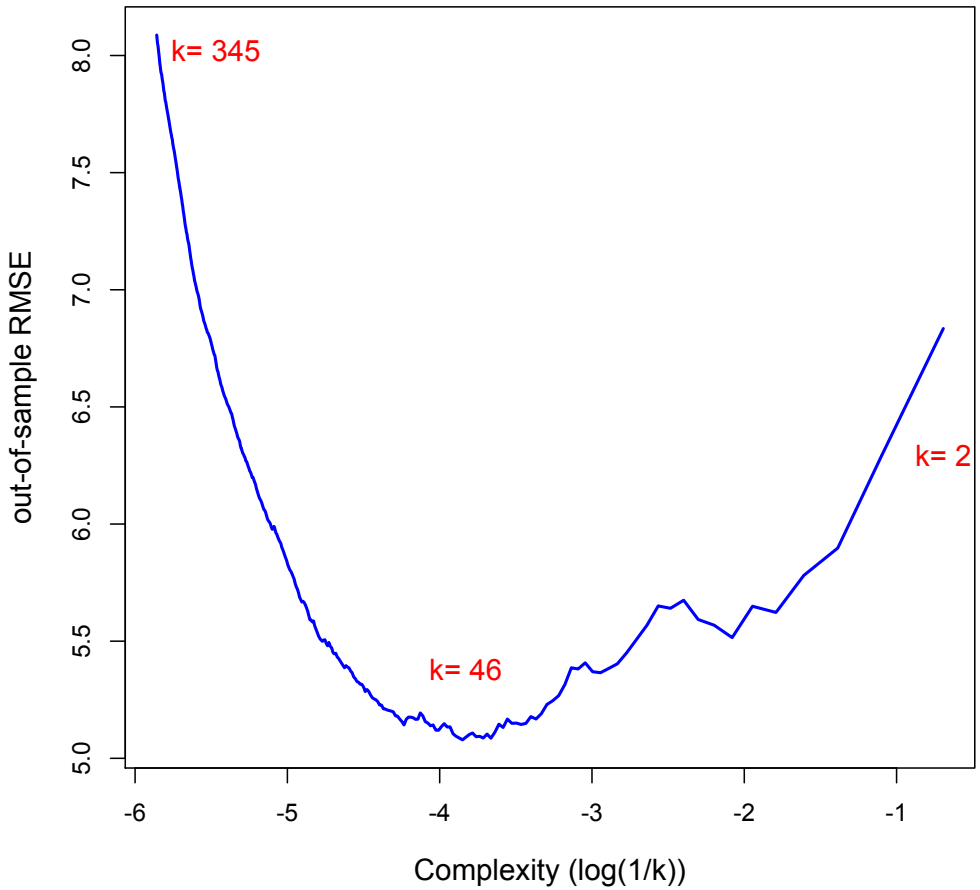
Suppose we have m additional observations (X_i^o, Y_i^o), for $i = 1, \ldots, m$, that we did not use to fit the model. Let's call this dataset the *validation set* (also known as *hold-out set* or *test set*)

Let's look at the out-of-sample RMSE:

$$RMSE^o = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[Y_i^o - \widehat{f(X_i^o)}\right]^2}$$

Out-of Sample Predictions

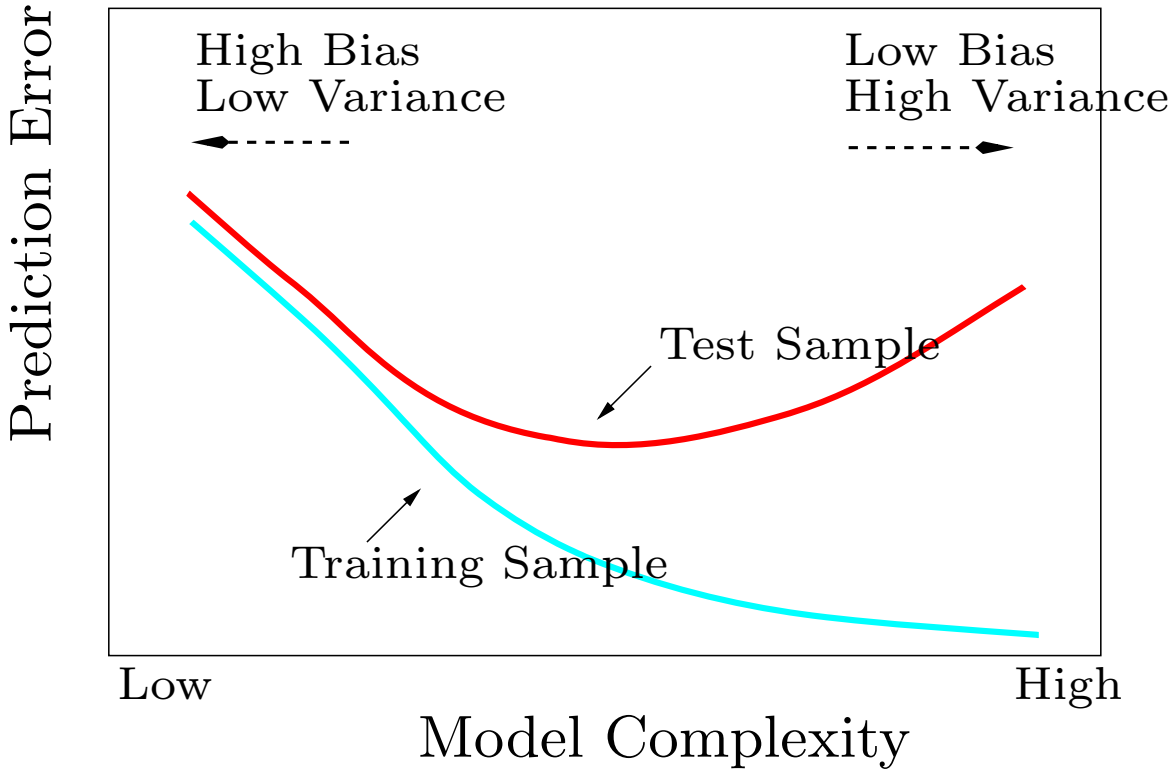
In our Boston housing example, I randomly chose a training set of size 400. I re-estimate the models using only this set and use the models to predict the remaining 106 observations (validation set)...



Now, the model where $k = 46$ looks like the most accurate choice!!

Not too simple but not too complex!!!

The Key Idea of the Course!!



This shows the typical behavior of the in and out-of-sample prediction error as a function of the complexity of the model... too flexible models will adapt itself too closely to the the training set and will not generalize well, i.e., not be very good for the test data.

4. Bias-Variance Trade-Off

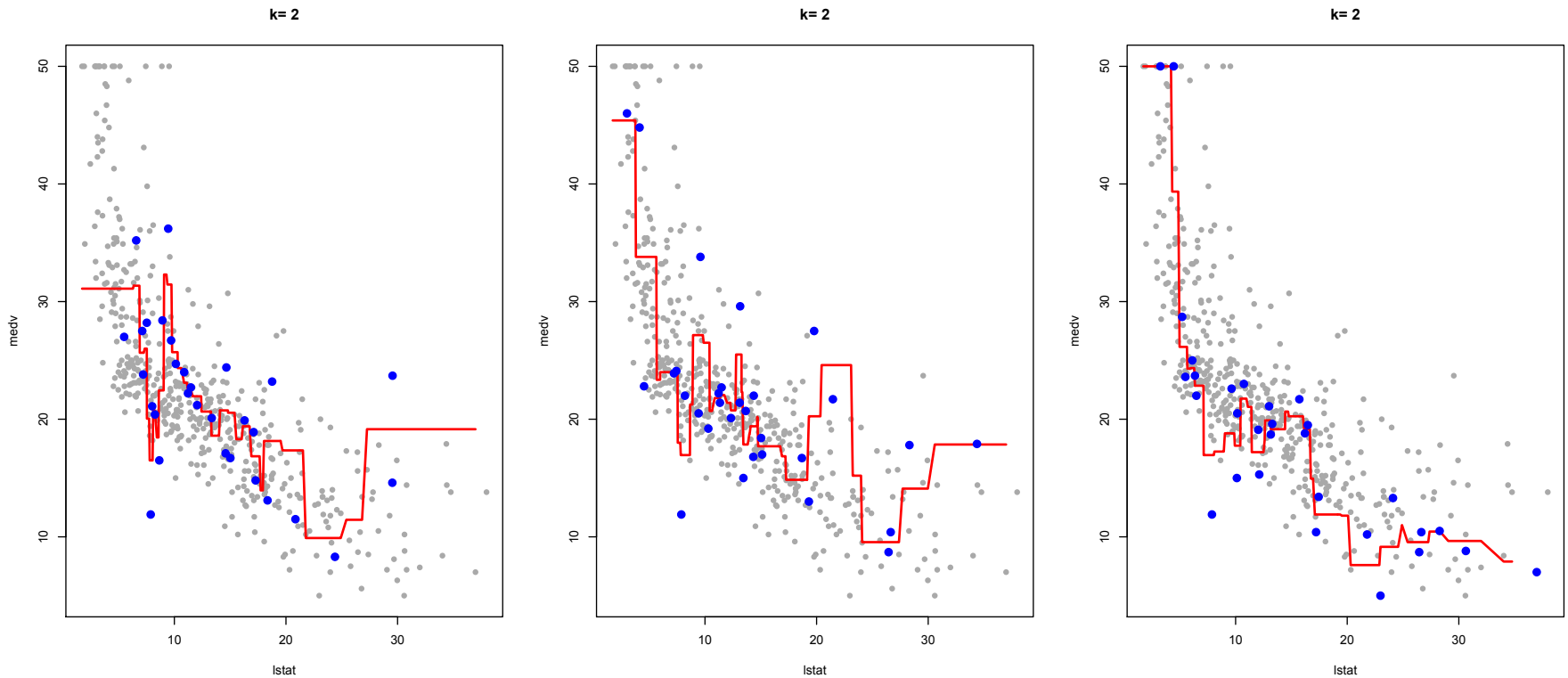
Why do complex models behave poorly in making predictions?

Let's start with an example...

- In the Boston housing example, I will randomly choose 30 observations to be in the training set 3 different times...
- for each training set I will estimate $f(\cdot)$ using the k-nearest neighbors idea... first with $k = 2$ and them with $k = 20$

Bias-Variance Trade-Off

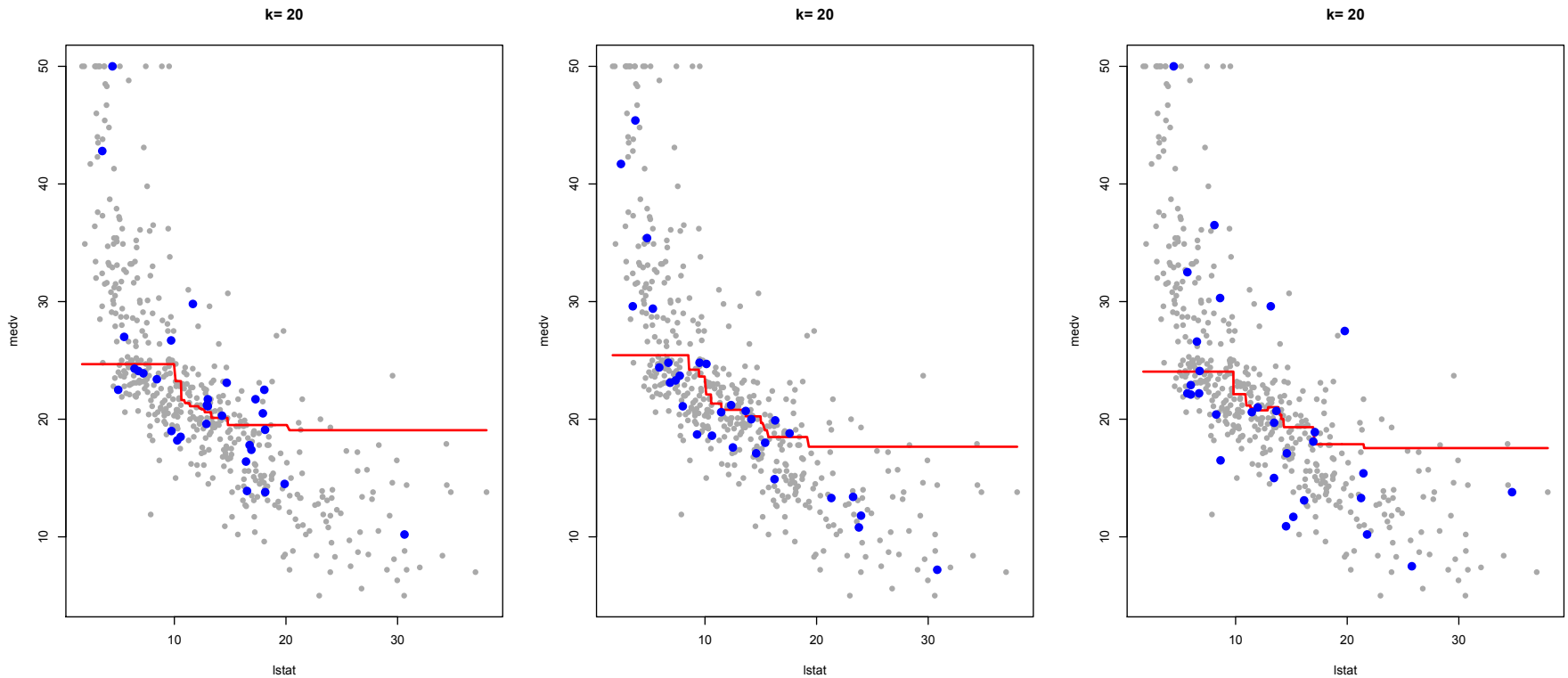
k=2 Hi variability...



(blue points are the training data used)

Bias-Variance Trade-Off

k=20 Low variability ... but BIAS!!



(blue points are the training data used)

Bias-Variance Trade-Off

What did we see here?

- When $k = 2$, it seems that the estimate of $f(\cdot)$ varies a lot between training sets...
- When $k = 20$ the estimates look a lot more stable...

Now, imagine that you are trying to predict *medv* when *lstat* $= 20$... compare the changes in the predictions made by the different training sets under $k = 2$ and $k = 20$... what do you see?

Bias-Variance Trade-Off

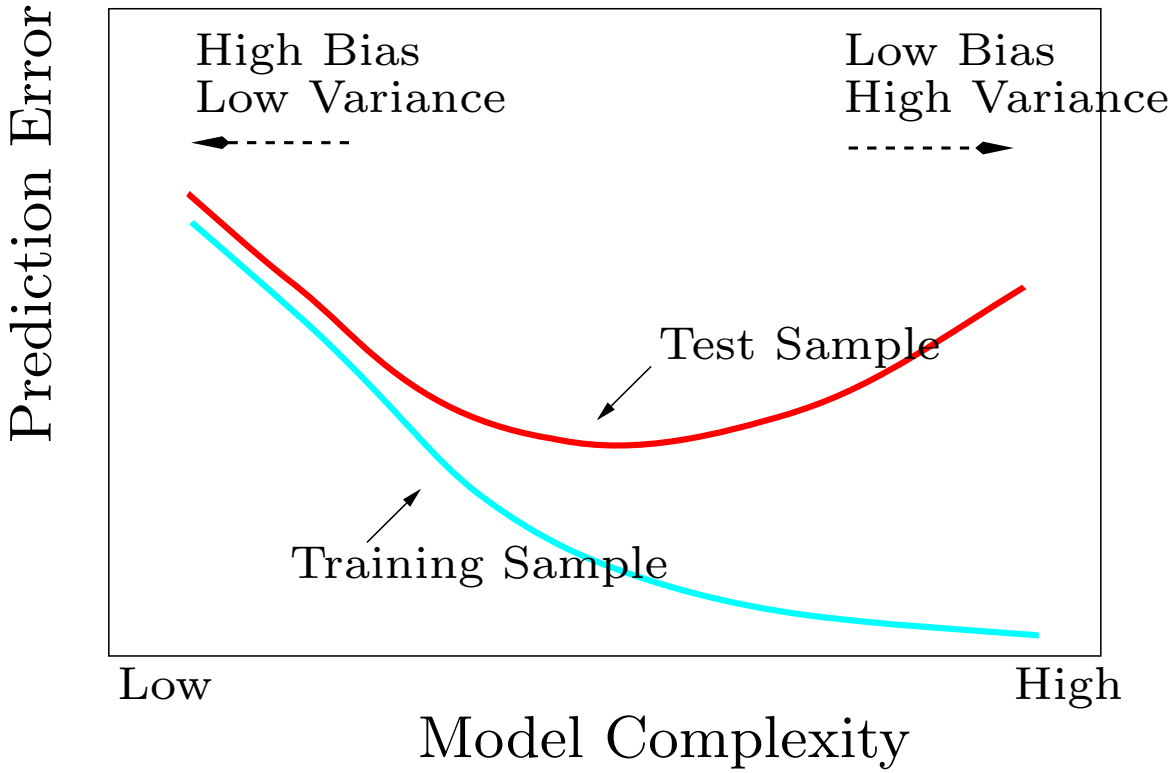
- This is an illustration of what is called the *bias-variance trade-off*.
- In general, simple models are trying to explain a complex, real problem with not a lot of flexibility so it introduces bias... on the other hand, by being simple the estimates tend to have low variance
- On the other hand, complex models are able to quickly adapt to the real situation and hence lead to small bias... however, by being to adaptable, it tends to vary a lot, i.e., high variance.

Bias-Variance Trade-Off

- In other words, we are trying to capture important patterns of the data that generalize to the future observations we are trying to predict. Models that are too simple are not able to capture relevant patterns and might have too big of a bias in predictions...
- Models that are too complex will "chase" irrelevant patterns in the training data that are not likely to exist in future data... so, it will lead to predictions that will vary a lot as things could change a lot depending on what sample we happen to see.
- Our goal is to find the sweet spot in the bias-variance trade-off!!

Bias-Variance Trade-Off

Once again, this is the key idea of the course!!



Bias-Variance Trade-Off

Let's get back to our original representation of the problem... it helps us understand what is going on...

$$Y_f = f(X_f) + \epsilon$$

- We need flexible enough models to find $f(\cdot)$ without imposing bias...
- ... but, too flexible models will "chase" non-existing patterns in ϵ leading to unwanted variability

Bias-Variance Trade-Off

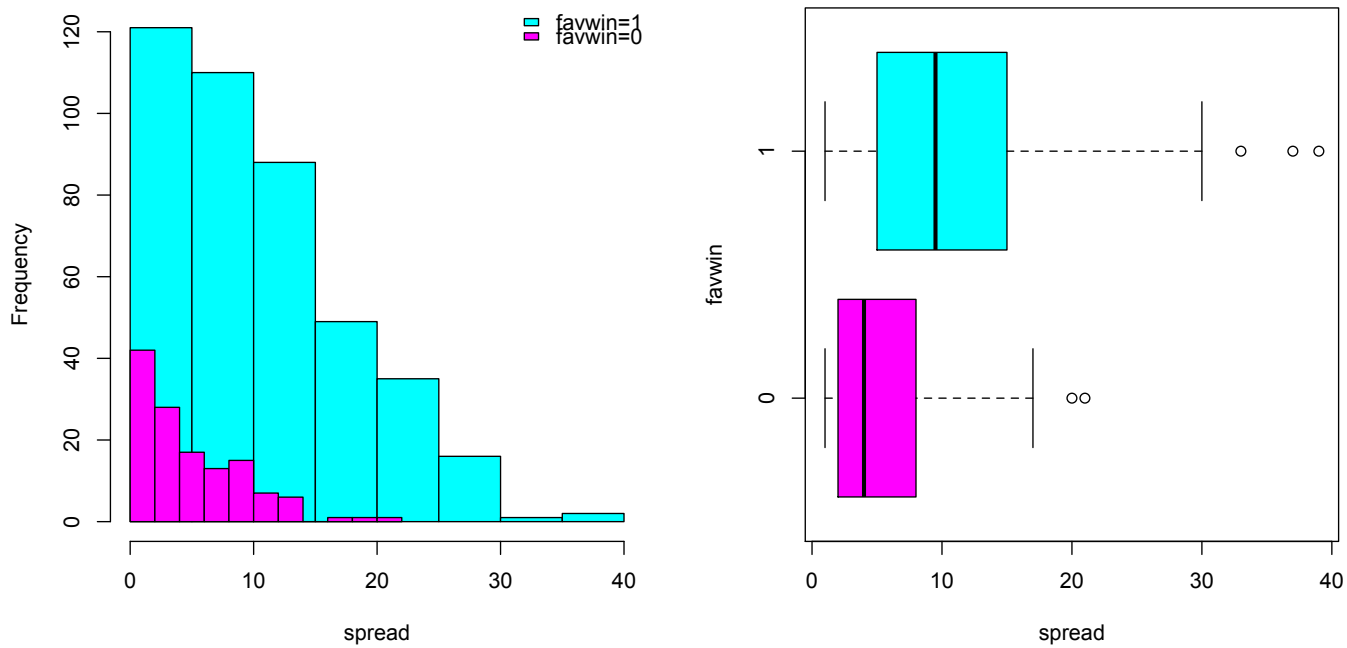
A more detailed look at this idea... assume we are trying to make a prediction for Y_f using X_f and our inferred $\widehat{f(\cdot)}$. We hope to make small mistake measured by squared distance... Let's explore how our mistakes will behave *on average*.

$$\begin{aligned} \mathsf{E}\left[(Y_f - \widehat{f(X_f)})^2\right] &= \left(f(X_f) - \mathsf{E}\left[\widehat{f(X_f)}\right]\right)^2 + \mathsf{Var}\left[\widehat{f(X_f)}\right] + \mathsf{Var}(\epsilon) \\ &= \mathsf{Bias}\left[\widehat{f(X_f)}\right]^2 + \mathsf{Var}\left[\widehat{f(X_f)}\right] + \mathsf{Var}(\epsilon) \end{aligned}$$

hence, the *Bias-Variance Trade-Off!!*

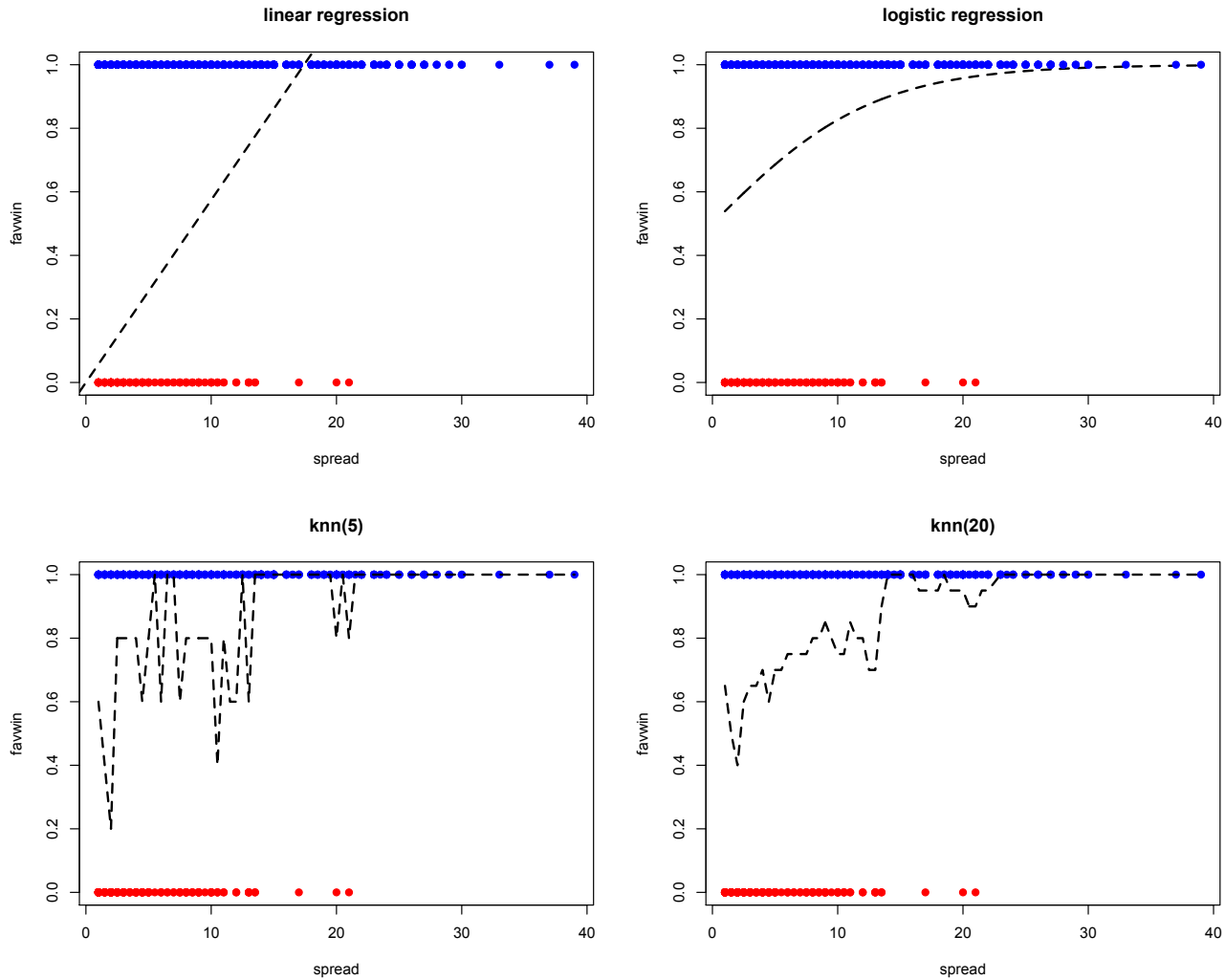
5. Classification

Classification is the term used when we want to predict a Y that is a category... win/lose, sick/healthy, pay/default, good/bad... below is an example where we are trying to predict whether or not the favorite team will win as a function of the Vegas' betting point-spread.



Classification

In this context, $f(X)$ will output the probability of Y taking a particular category (win/lose,here). Below, the black line represent different methods to estimate $f(X)$ in this example.



Classification

All the discussion about complexity vs. predictive accuracy (bias-variance trade-off) applies to this setting... what differs is our measure of accuracy as we are no longer dealing with a numerical outcome variable. A common approach is to measure the *error rate*, i.e., the number of times we a estimated $\widehat{f(\cdot)}$ "gets it wrong"... in the training set (with n observations) define the error rate as:

$$\frac{1}{n}\sum_{i=1}^{n} \mathsf{I}(Y_i \neq \hat{Y}_i)$$

where $\hat{Y}_i$ is the class (category) label assigned by $\widehat{f(X_i)}$ and $\mathsf{I}(\cdot)$ is an indicator variable that equals 1 whenever $Y_i \neq \hat{Y}_i$ and 0 otherwise.

Classification

As mentioned above, in classification, $\widehat{f(X_i)}$ outputs a probability. In order to use this information and create a *classification rule* we need to decide on a cut-off (or cut-offs) to determined the label assignments.

In general, if we are looking at only two categories (like the NBA example above) the standard cut-off is 0.5. This means that if $\widehat{f(X_f)} > 0.5$ we define $\hat{Y}_i = 1$ ($\hat{Y}_i = 0$ otherwise).

Later on, we will think more carefully about defining cut-offs...

Classification

- Extending this metric to the testing set is straightforward. For m samples (X_i^o, Y_i^o) the error rate is: $\frac{1}{m}\sum_{i=1}^{n} \mathsf{I}(Y_i^o \neq \hat{Y}_i^o)$
- In general, we need to think carefully about choosing the cut-off and defining a classification rule. In the NBA example above, what is a natural cut-off?
- The error rate treats a "mistake" equally, regardless of its type. Is that always a good idea? Think about deciding if a defender is guilty or not?
- We will explore all of these issues more carefully when we get to the classification section. For now, let's keep the error rate as our accuracy measure.

6. Cross-Validation

- Using a validation-set to evaluate the performance of competing models has two potential drawbacks:
  1. the results can be highly dependent on the choice of the validation set... what samples? how many?
  2. by leaving aside a subset of data for validation we end up estimating the models with less information. It is harder to *learn* with fewer samples and this might lead to an overestimation of errors.
- Cross-Validation is a refinement of the validation strategy that help address both of these issues.

Leave-One-Out Cross-Validation (loocv)

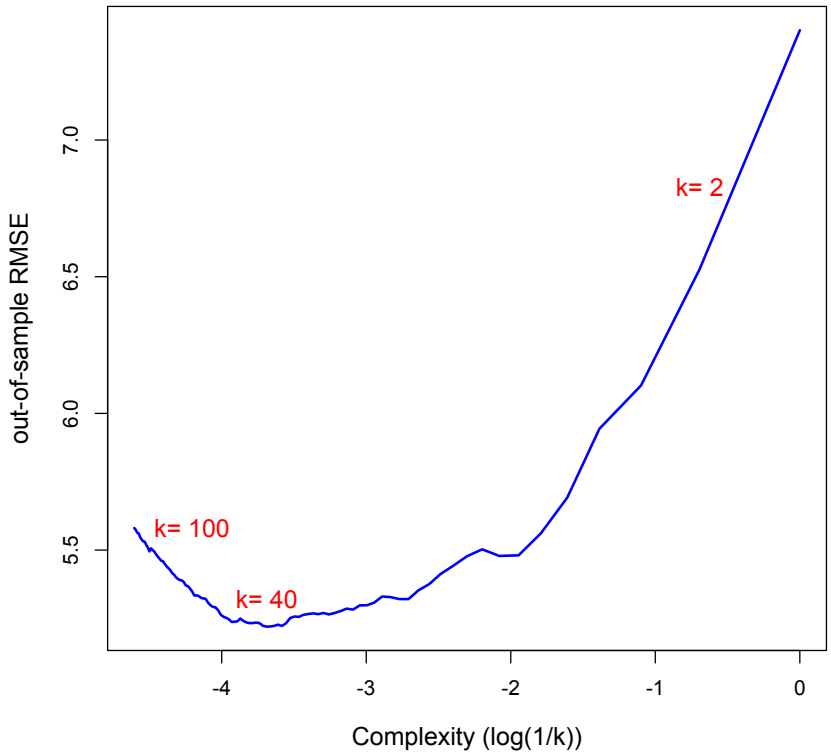
The name says it all!

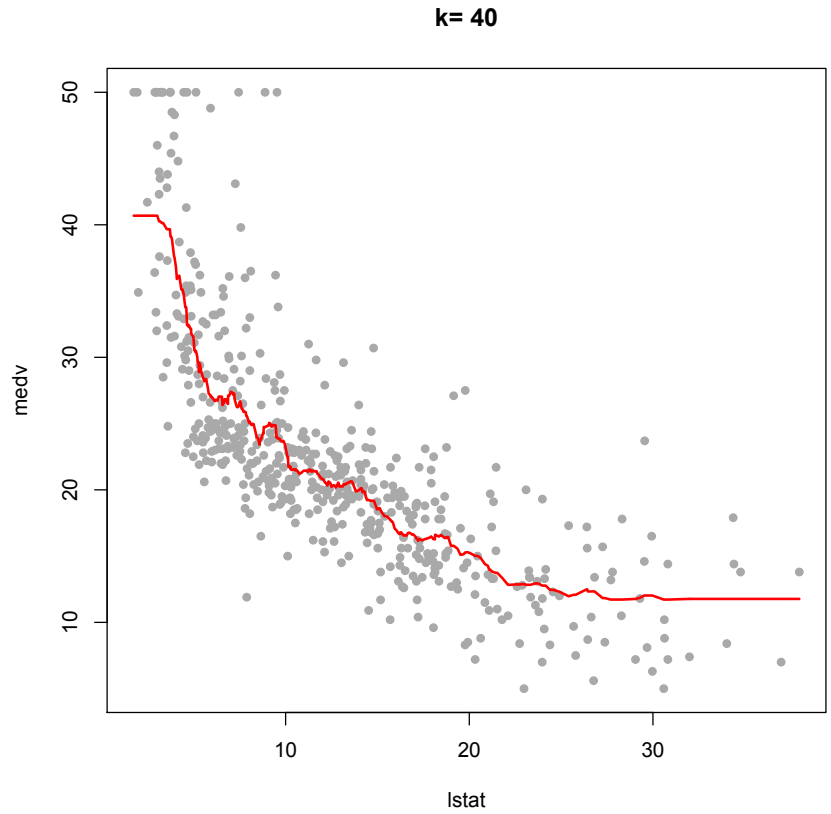
- Assume we have n samples in our dataset. Define the validation set by choosing only one sample. Call it the i^{th} observation...
- The model is then trained in the remaining $n-1$ samples and the results are used to predict the left-out sample. Compute the squared-error $MSE_i = (Y_i - \hat{Y}_i)^2$
- Repeat the procedure for every sample in the dataset (n times) and compute the average cross-validation MSE:

$$MSE^{loocv} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

Leave-One-Out Cross-Validation

In the Boston data....





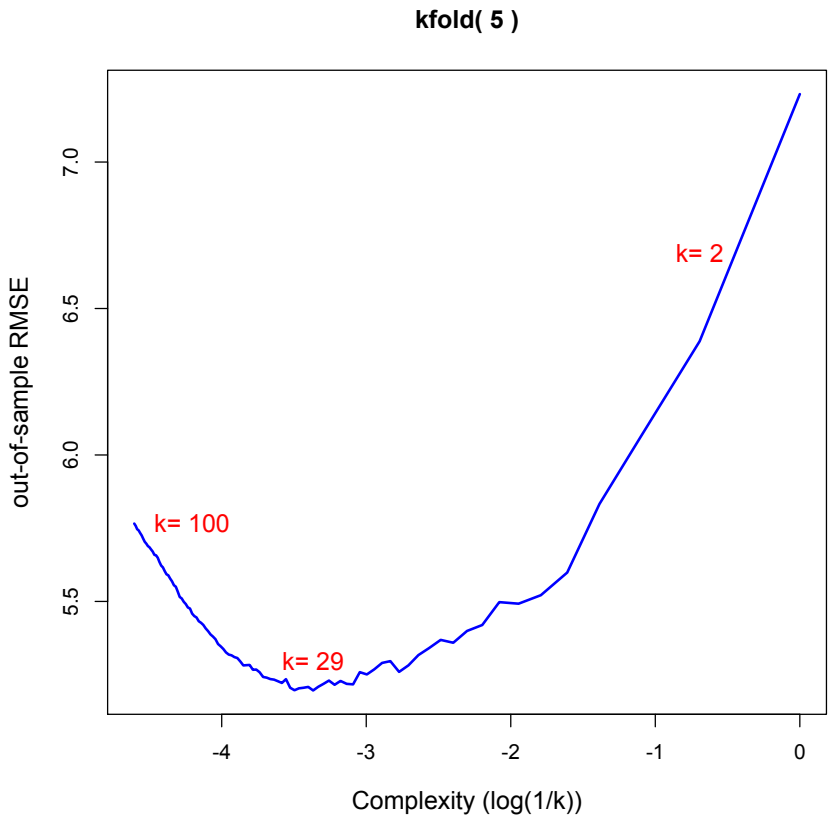
k-fold Cross Validation

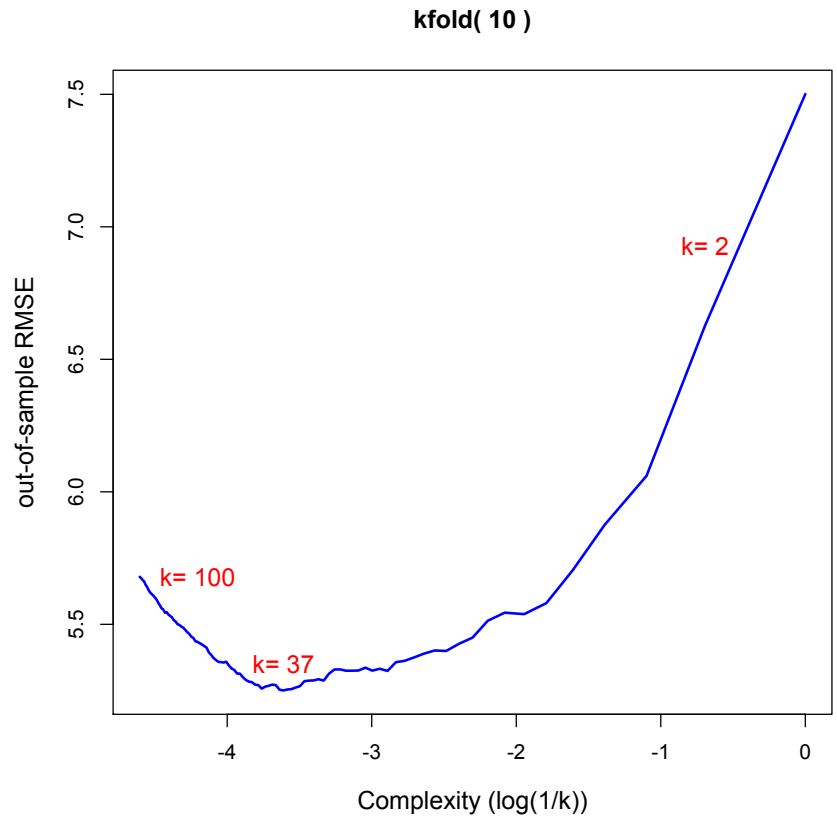
LOOCV can be computationally expensive as each model being considered has to be estimated n times! A popular alternative is what is called k-fold Cross Validation.

- This approach randomly divides the original dataset into k groups of approximately the same size
- Choose one of the groups as a validation set. Estimate the models with the remaining $k-1$ groups and predict the samples in the validation set. Compute the average squared-error for the samples in the validation set
  $MSE_i = \text{Average}\left[(Y_i - \hat{Y}_i)^2\right]$
- Repeat the procedure for every *fold* in the dataset (k times) and compute the average cross-validation MSE:
  $MSE^{kcv} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$

k-fold Cross Validation

The usual choices are between $k = 5$ and $k = 10$...





k-fold Cross Validation

- Bias-Variance Trade-Off: LOOCV has a smaller bias as it uses more information in the training sets... but it also has more variance! By only leaving a sample out at a time, the training set doesn't change much so the outputs will be very correlated with each other. So, the resulting MSE will be a average off correlated quantities which tends to be more variable
- In k-fold the outputs are not very correlated as the training sets can be quite different and the resulting MSE is an average of somewhat "independent" quantities... hence less variance!
- There is a bias-variance trade-off between LOOCV and k-fold. The choice of $k = 5$ or $k = 10$ has been empirically shown to be a good choice that neither suffers from a lot of bias nor from a lot of variance!

7. k-Nearest Neighbors (kNN)

The k-nearest neighbors algorithm will try to *predict* (numerical variables) or *classify* (categorical variables) based on similar (close) records on the *training dataset*.

Remember, the problem is to guess a future value Y_f given new values of the covariates $X_f = (x_{1f}, x_{2f}, x_{3f}, \ldots, x_{pf})$.

k-Nearest Neighbors (kNN)

kNN: How do the $Y's$ look like close to the region around X_f?

We need to find the k records in the training dataset that are close to X_f. How? "Nearness" to the i^{th} neighbor can be defined by (euclidian distance):

$$d_i = \sqrt{\sum_{j=1}^{p}(x_{jf} - x_{ji})^2}$$

Prediction:

- Numerical Y_f: take the average of the $Y's$ in the k-nearest neighbors
- Categorical Y_f: take the most common category in the k-nearest neighbors

k-Nearest Neighbors (kNN) – Example

Forensic Glass Analysis

Classifying shards of glass

Refractive index, plus oxide %

Na, Mg, Al, Si, K, Ca, Ba, Fe.

6 possible glass types

`WinF:` float glass window

`WinNF:` non-float window

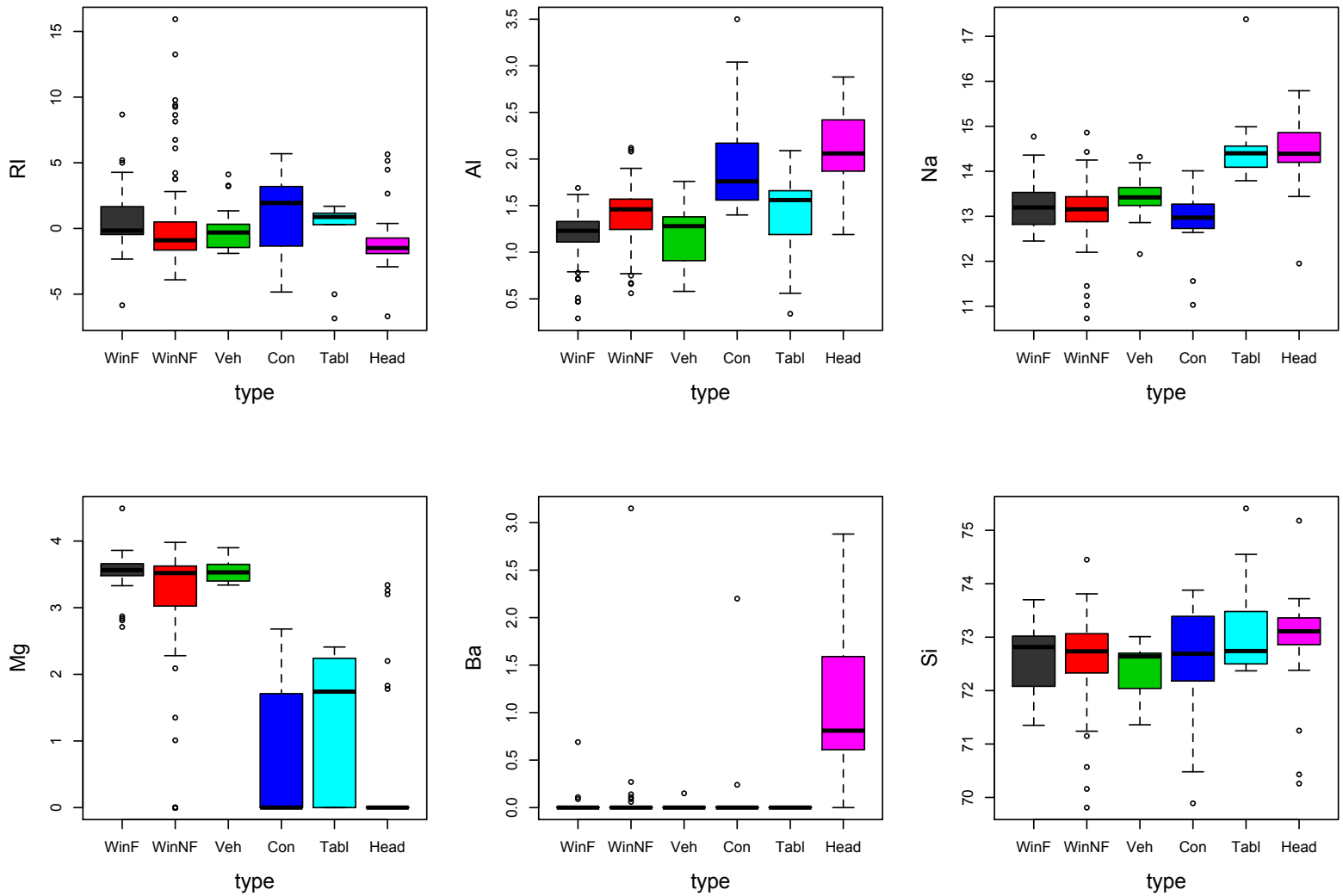
`Veh:` vehicle window

`Con:` container (bottles)

`Tabl:` tableware

`Head:` vehicle headlamp

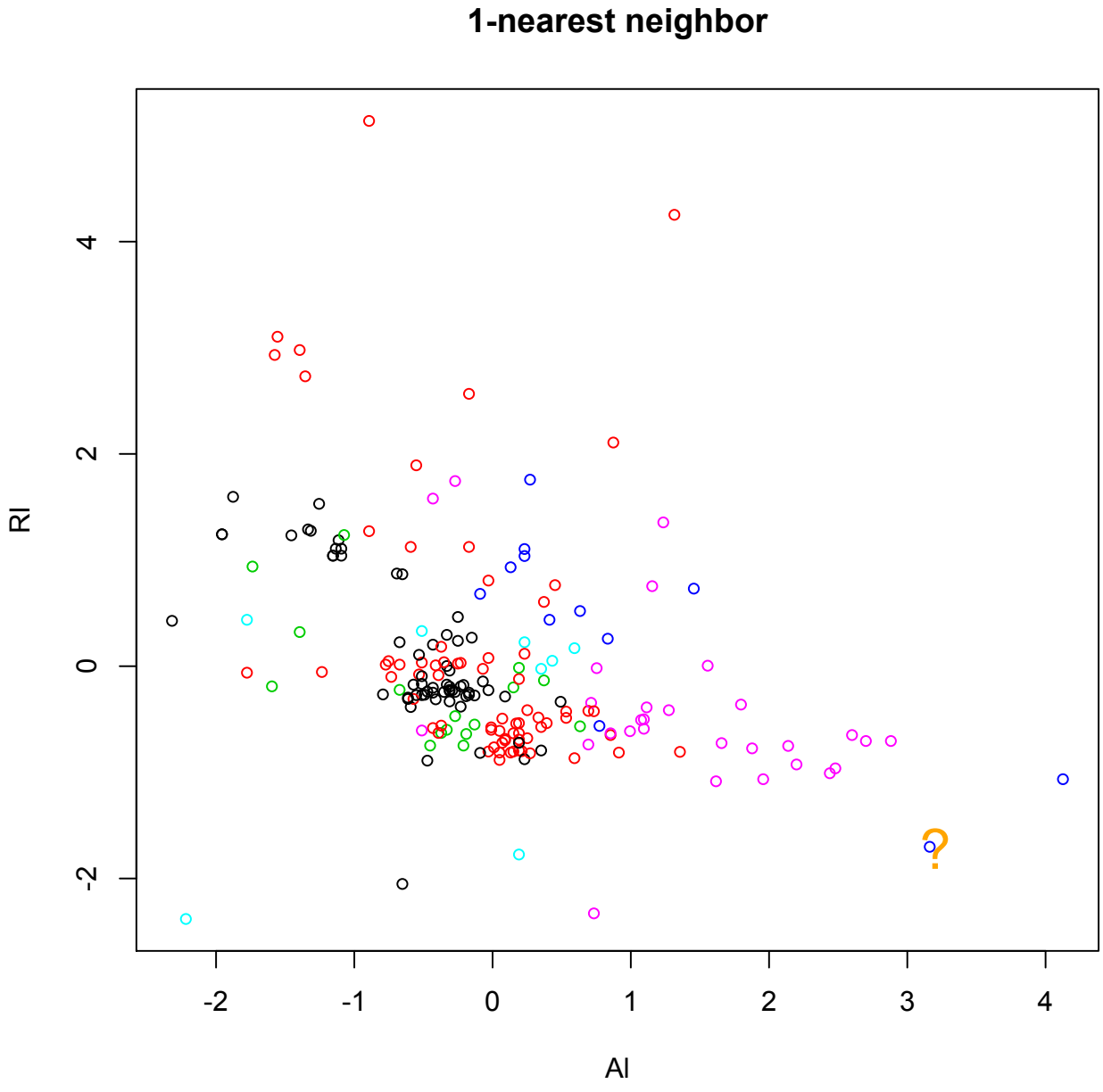
k-Nearest Neighbors (kNN) – Example



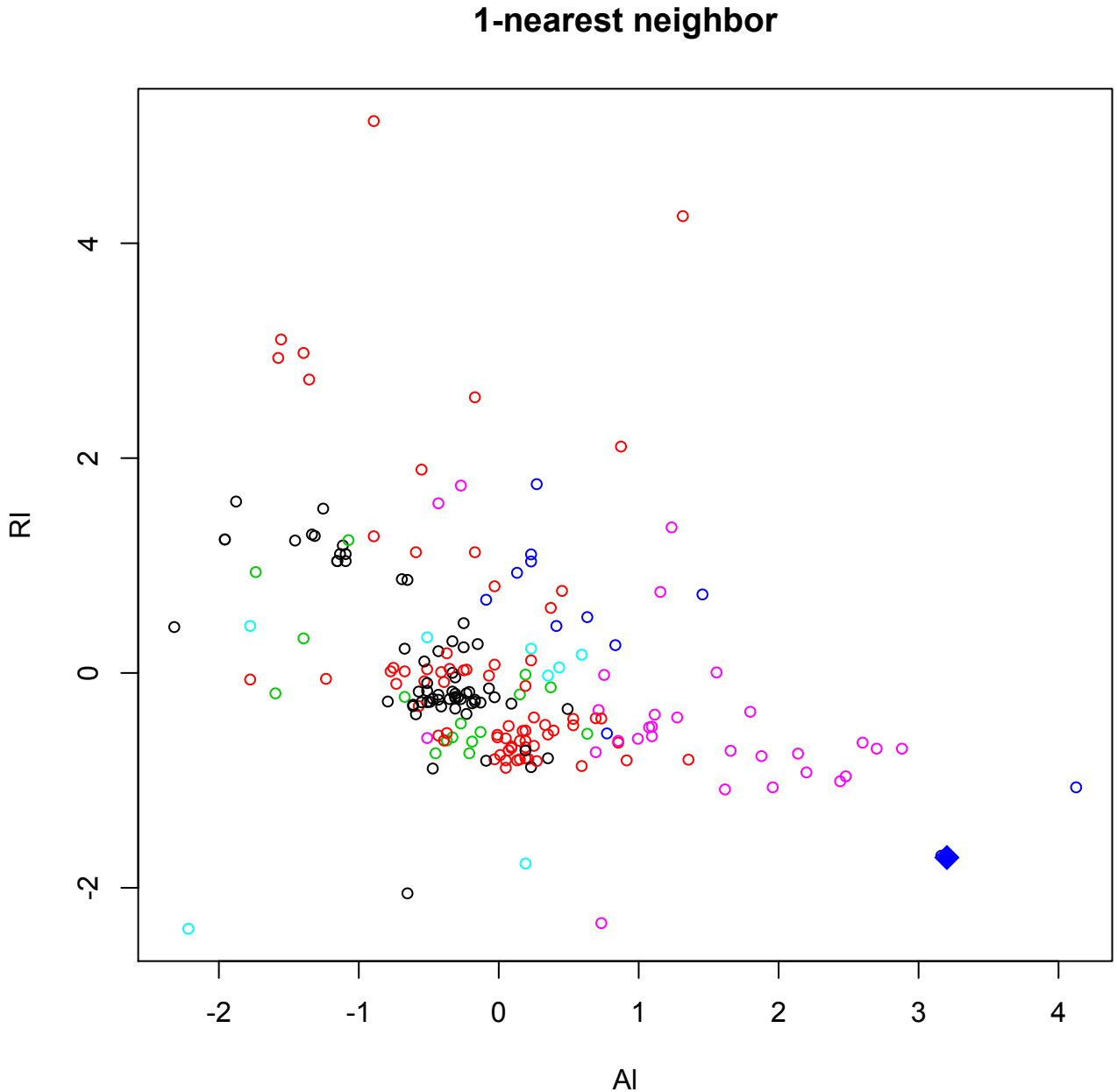
Some variables are clear discriminators (Ba) while others are more subtle (RI)

k-Nearest Neighbors (kNN) – Example

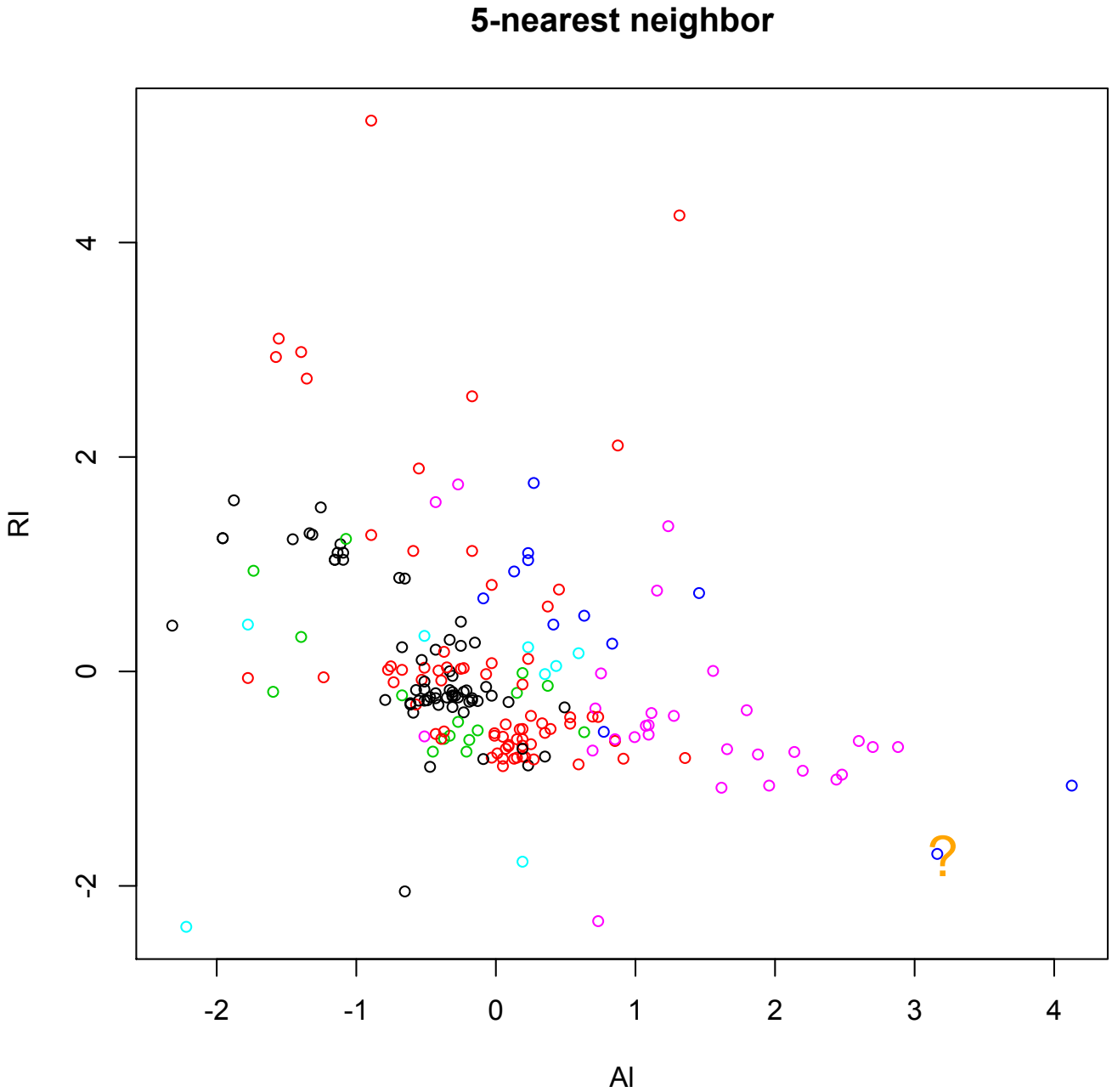
Using only RI and Al let's try to predict the point marked by "?"



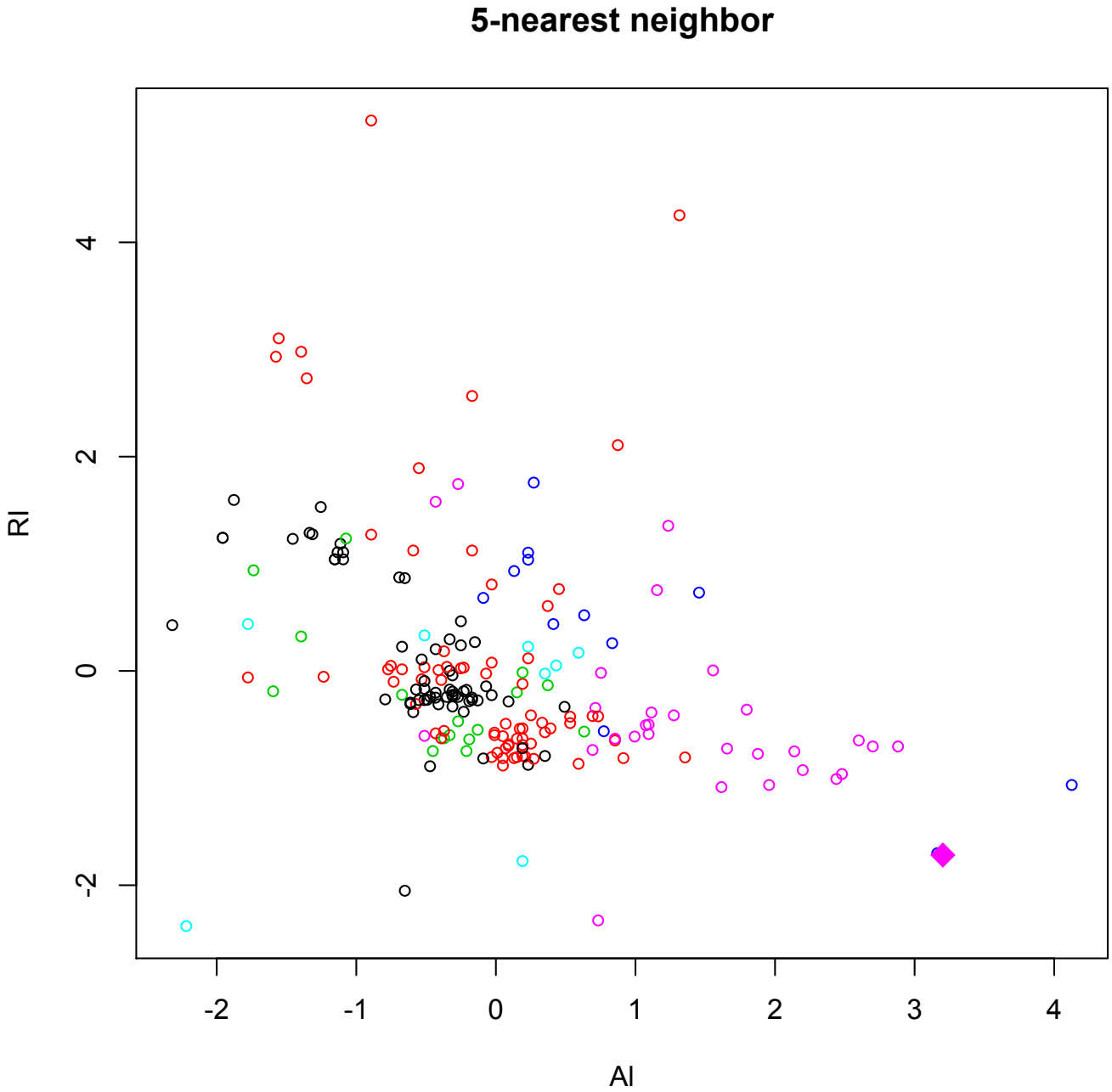
k-Nearest Neighbors (kNN) – Example



k-Nearest Neighbors (kNN) – Example



k-Nearest Neighbors (kNN) – Example



k-Nearest Neighbors (kNN)

Comments:

- kNN is simple and intuitive and yet very powerful! e.g. Pandora (music genome project), Nate Silver's first company...
- Choice of k matters! Once again, we should rely on the out-of-sample performance
- As always, deciding the cut-off for classification impacts the results (more on this later)

k-Nearest Neighbors (kNN)

More comments:

- The distance metric used above is only valid for numerical values of X. When $X's$ are categorical we need to think about a different distance metric or perform some manipulation of the information.
- The scale of X also will have an impact. In general it is a good idea put the $X's$ in the same scale before running kNN (see example in `R`)

knn: California Housing

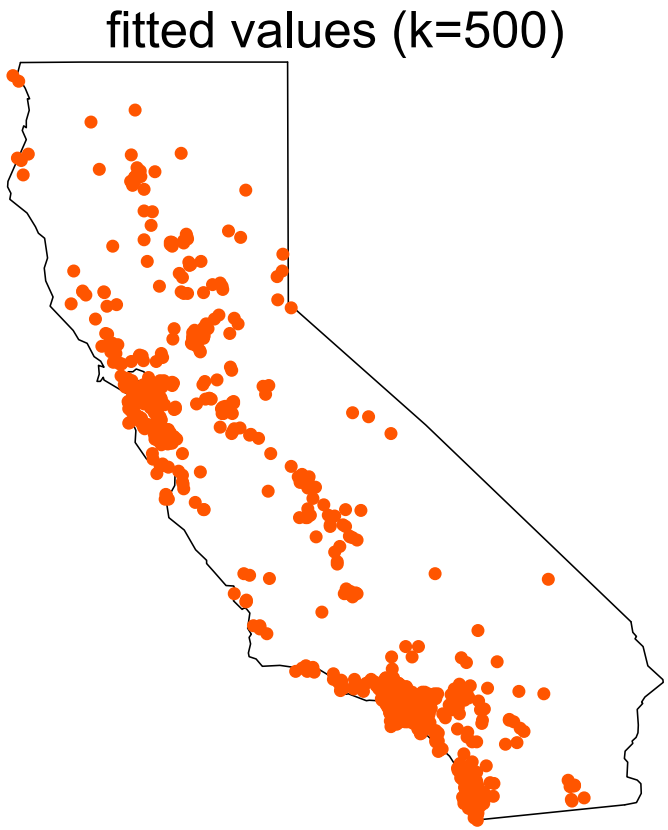
Data: Medium home values in census tract plus the following information:

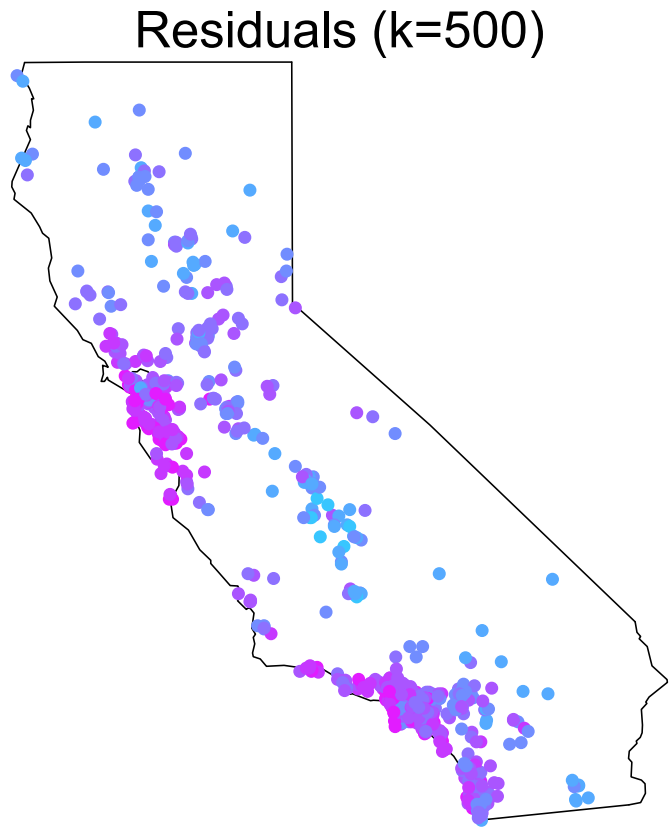
- Location (latitude, longitude)
- Demographic information: population, income, etc...
- Average room/bedroom number, home age
- Let's start using just location as our $X's$... euclidian distance is quite natural here, right?

Goal: Predict $log(MediumValue)$ (why logs? more on this later)

knn: California Housing

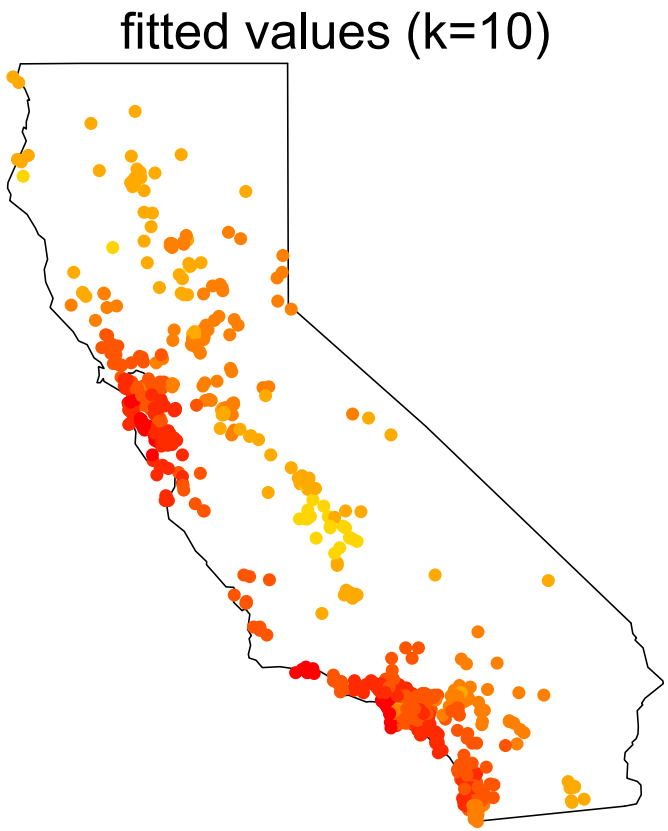
Using a training data of $n = 1000$ samples, here's a picture of the results for $k = 500$... left $\hat{Y}_i$, right $(Y_i - \hat{Y}_i)$

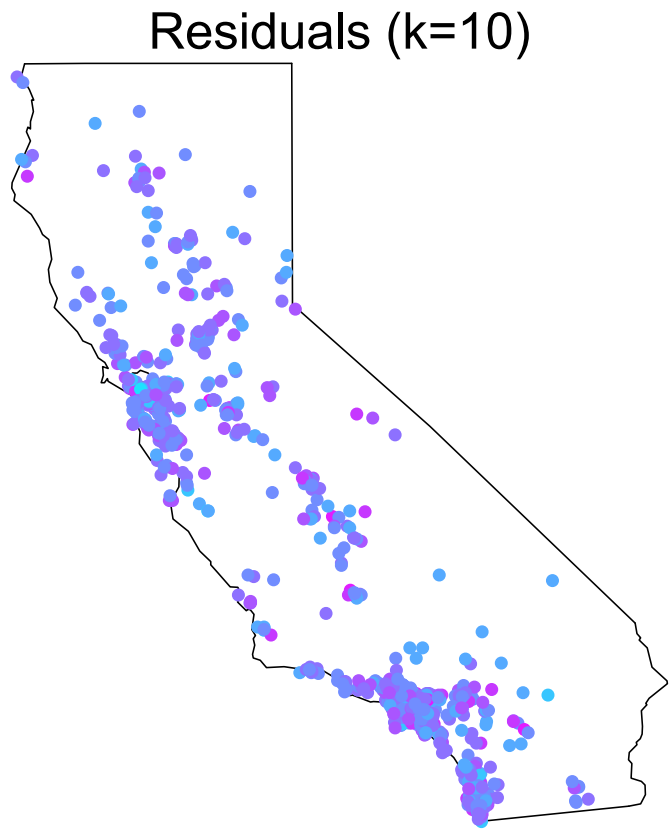




knn: California Housing

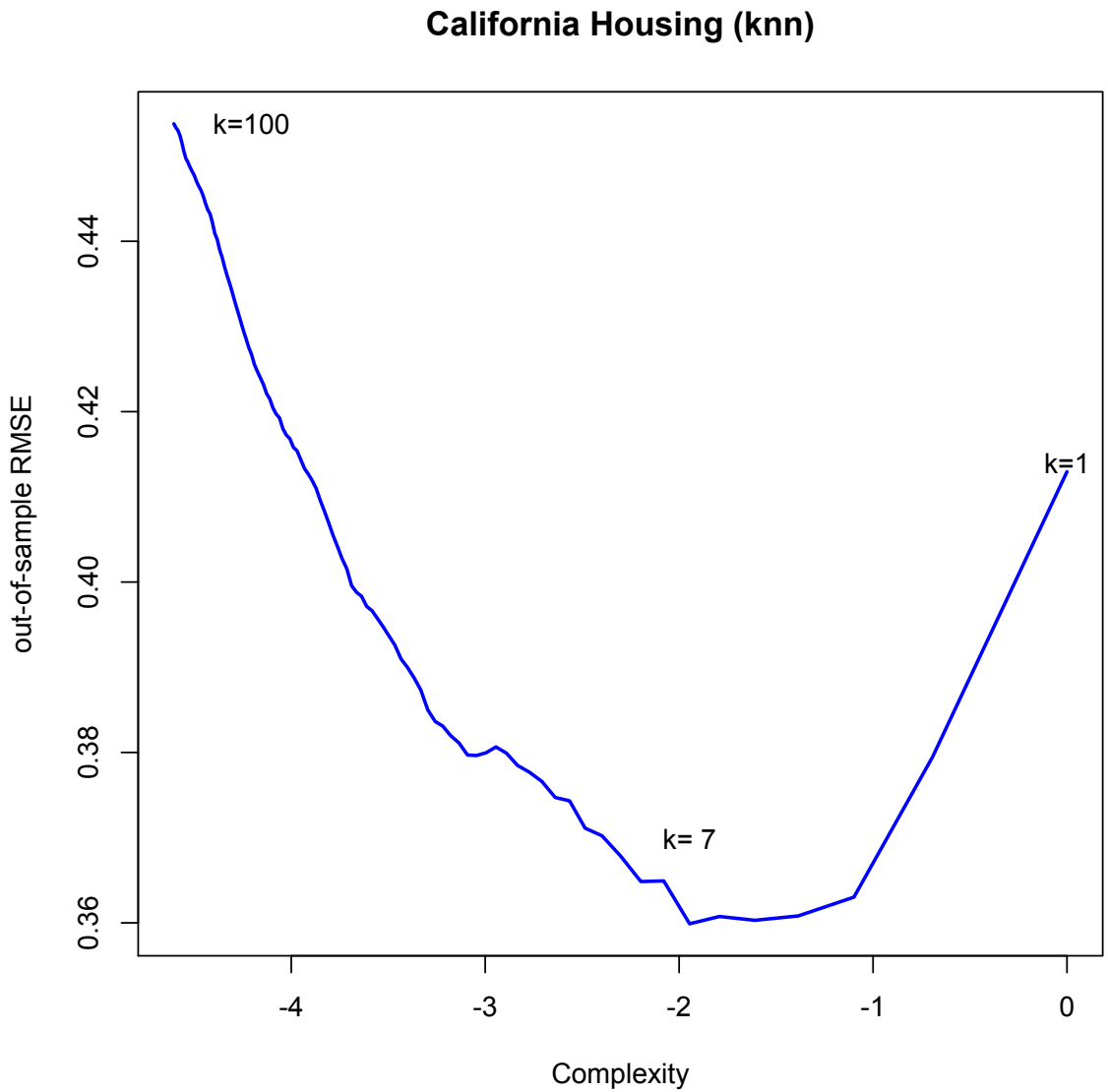
Now, $k = 10$... does this make sense?





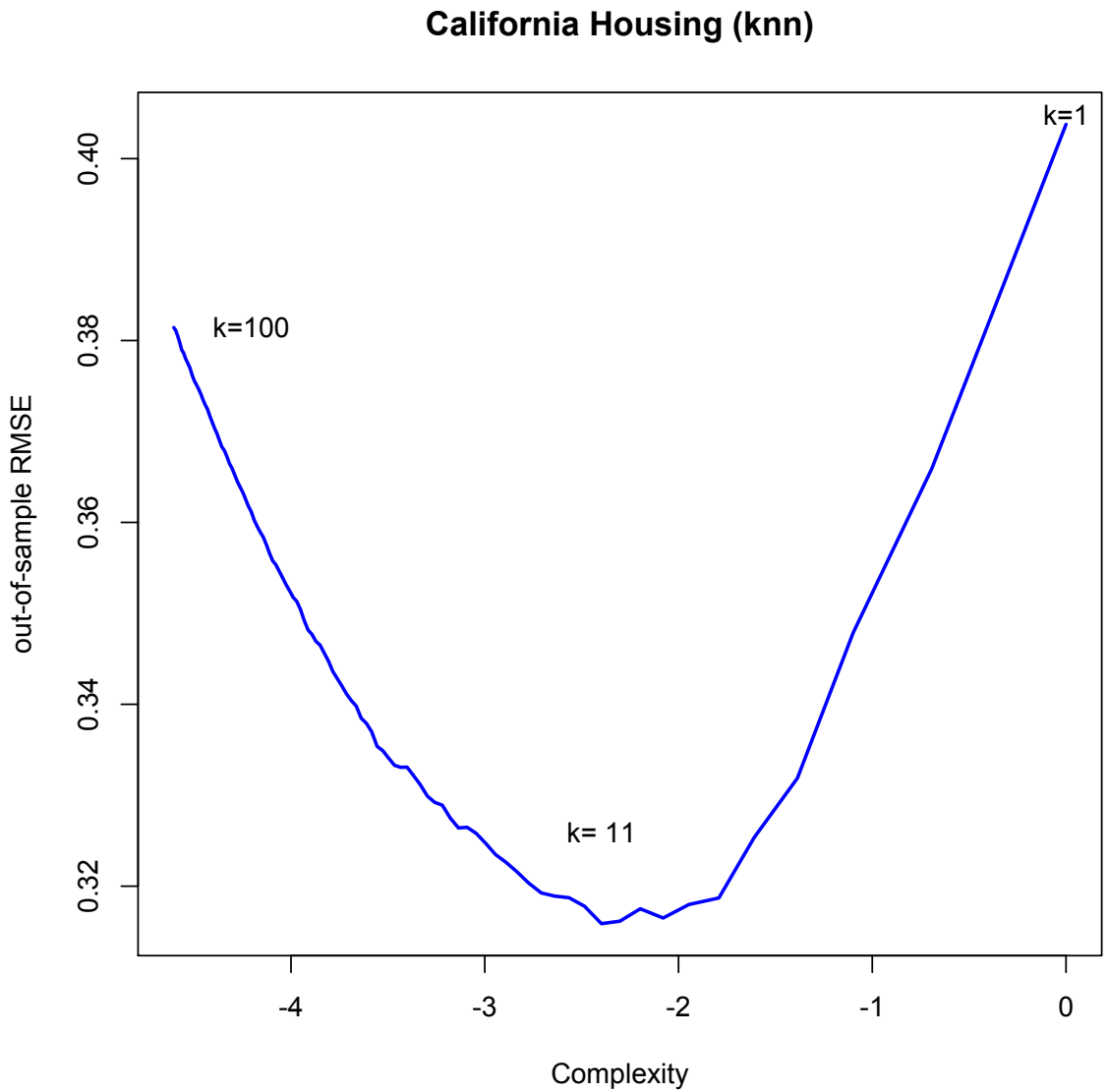
knn: California Housing

k-fold cross validation... $kfold = 10$



knn: California Housing

Adding Income as a predictor... think about the scale of X...



knn: California Housing

Adding Income as a predictor... think about the scale of X...