# Lecture 12: *k*-NN Classification and Dealing with Missingness

## CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner

## Lecture Outline

- *k*-NN review
- *k*-NN for Classification
- Dealing with Missingness
    - Types of Missingness
    - Imputation Methods

## *k*-Nearest Neighbors

We've already seen the *k*-NN method for predicting a quantitative response (it was the very first method we introduced). How was *k*-NN implemented in the Regression setting (quantitative response)?

The approach was simple: to predict an observation's response, use the **other** available observations that are most similar to it.

For a specified value of *k*, each observation's outcome is predicted to be the **average** of the *k*-closest observations as measured by some distance of the predictor(s).

With one predictor, the method was easily implemented.

## Review: Choice of k

How well the predictions perform is related to the choice of *k*.

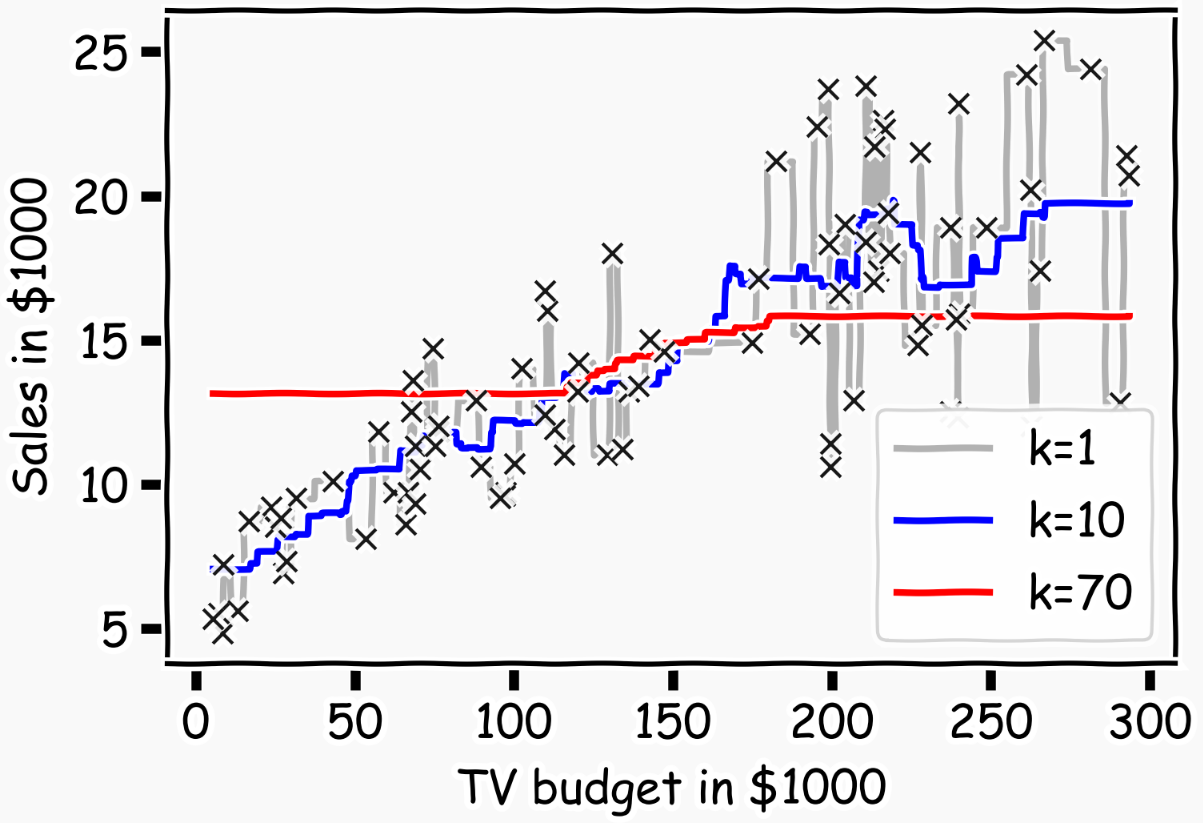
What will the predictions look like if *k* is very small? What if it is very large?

More specifically, what will the predictions be for new observations if $k = n$?

$$\bar{Y}$$

A picture is worth a thousand words...

# Choice of *k* matters



## *k*-NN for Classification

## *k*-NN for Classification

How can we modify the *k*-NN approach for classification?

The approach here is the same as for *k*-NN regression: use the other available observations that are most similar to the observation we are trying to predict (classify into a group) based on the predictors at hand.

How do we classify which category a specific observation should be in based on its nearest neighbors?

The category that shows up the most among the nearest neighbors.

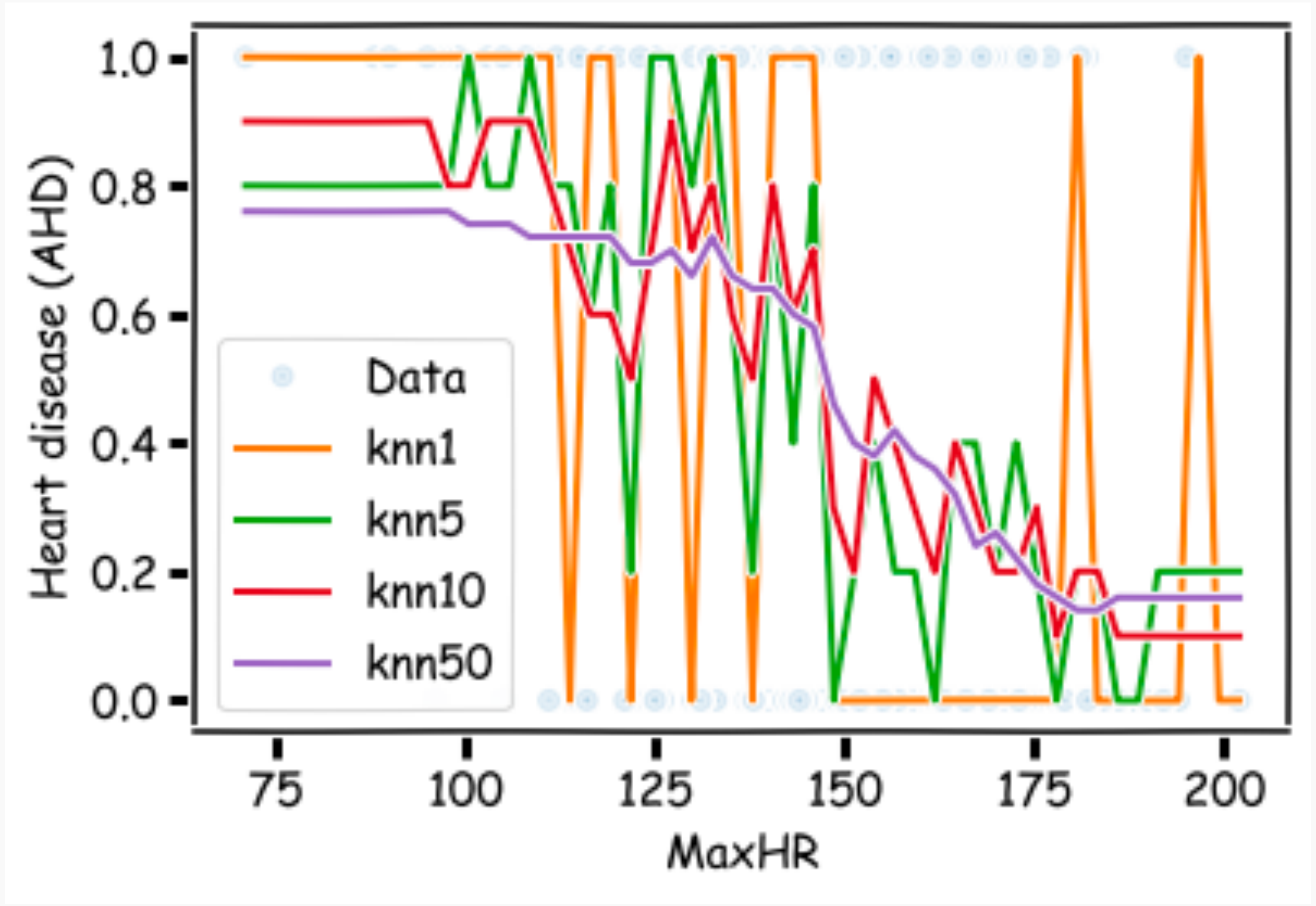
## *k*-NN for Classification: formal definition

The *k*-NN classifier first identifies the *k* points in the training data that are closest to $x_0$, represented by $\mathcal{N}_0$. It then estimates the conditional probability for class *j* as the fraction of points in $\mathcal{N}_0$ whose response values equal *j*:

$$P(Y = j | X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

Then, the *k*-NN classifier applies Bayes rule and classifies the test observation, $x_0$, to the class with largest estimated probability.

## Estimated Probabilities in *k*-NN Classification



## *k*-NN for Classification (cont.)

There are some issues that may arise:

- How can we handle a tie? With a coin flip!
- What could be a major problem with always classifying to the most common group amongst the neighbors?

  If one category is much more common than the others then all the predictions may be the same!
- How can we handle this?

  Rather than classifying with the most likely group, use a biased coin flip to decide which group to classify to.

## *k*-NN with Multiple Predictors

How could we extend *k*-NN (both regression and classification) when there are multiple predictors?

We would need to define a measure of distance for observations in order to which are the most similar to the observation we are trying to predict.

Euclidean distance is a good option. To measure the distance of a new observation, $\boldsymbol{x}_0$ from each observation in the data set, $\boldsymbol{x}_i$:

$$D^2(\boldsymbol{x}_i, \boldsymbol{x}_0) = \sum_{j=1}^{P} \left(x_{i,j} - x_{0,j}\right)^2$$

## *k*-NN with Multiple Predictors (cont.)

But what must we be careful about when measuring distance?

1. Differences in variability in our predictors!
2. Having a mixture of quantitative and categorical predictors.

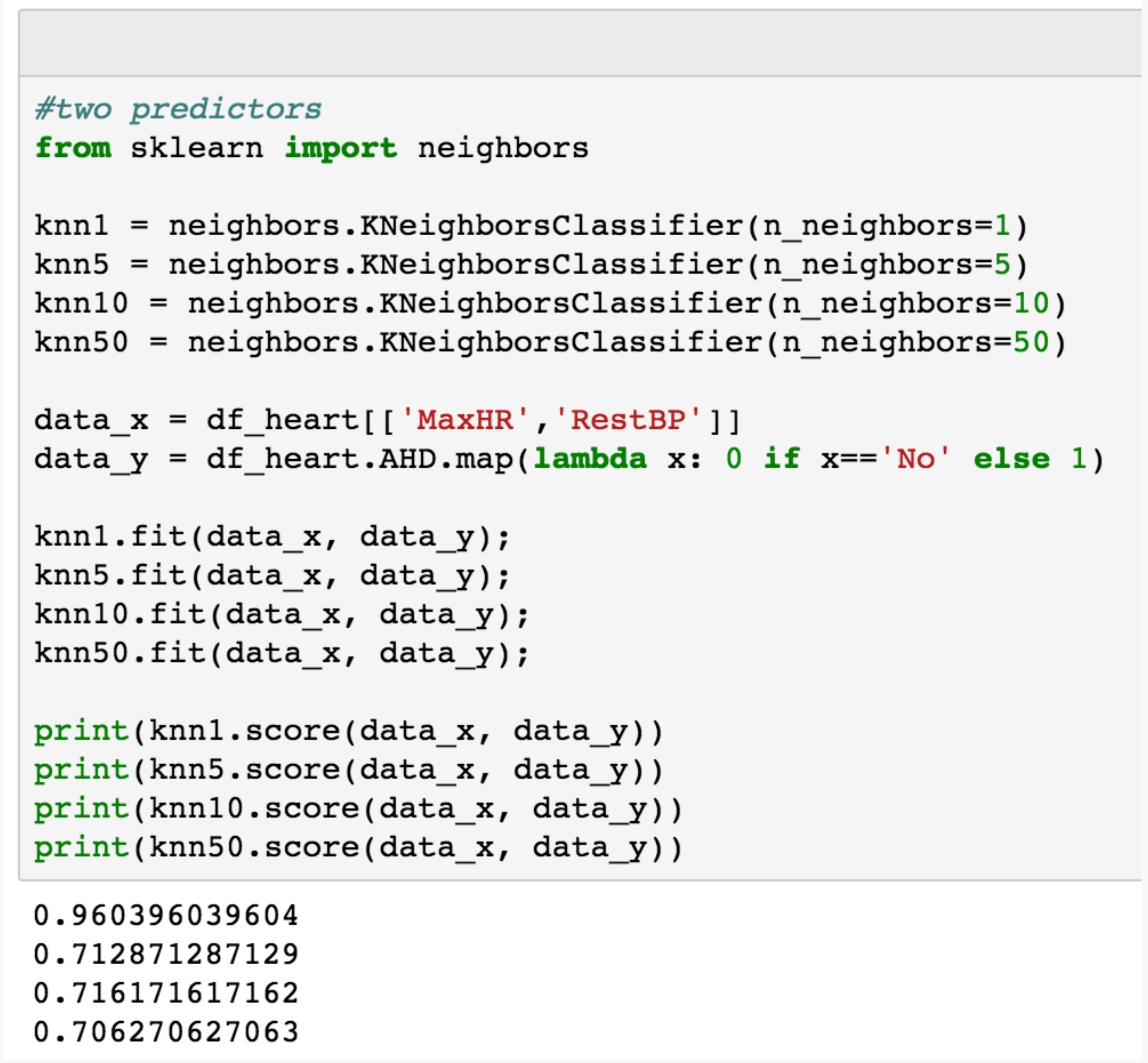
So what should be good practice? To determine closest neighbors when $p > 1$, you should first standardize the predictors! And you can even standardize the binaries if you want to include them.

How else could we determine closeness in this multi-dimensional setting?

# *k*-NN Classification in Python

Performing kNN classification in python is done via **`KNeighborsClassifier`** in `sklearn.`**`neighbors`**`.`

An example:



```python
#two predictors
from sklearn import neighbors

knn1 = neighbors.KNeighborsClassifier(n_neighbors=1)
knn5 = neighbors.KNeighborsClassifier(n_neighbors=5)
knn10 = neighbors.KNeighborsClassifier(n_neighbors=10)
knn50 = neighbors.KNeighborsClassifier(n_neighbors=50)

data_x = df_heart[['MaxHR','RestBP']]
data_y = df_heart.AHD.map(lambda x: 0 if x=='No' else 1)

knn1.fit(data_x, data_y);
knn5.fit(data_x, data_y);
knn10.fit(data_x, data_y);
knn50.fit(data_x, data_y);

print(knn1.score(data_x, data_y))
print(knn5.score(data_x, data_y))
print(knn10.score(data_x, data_y))
print(knn50.score(data_x, data_y))
```

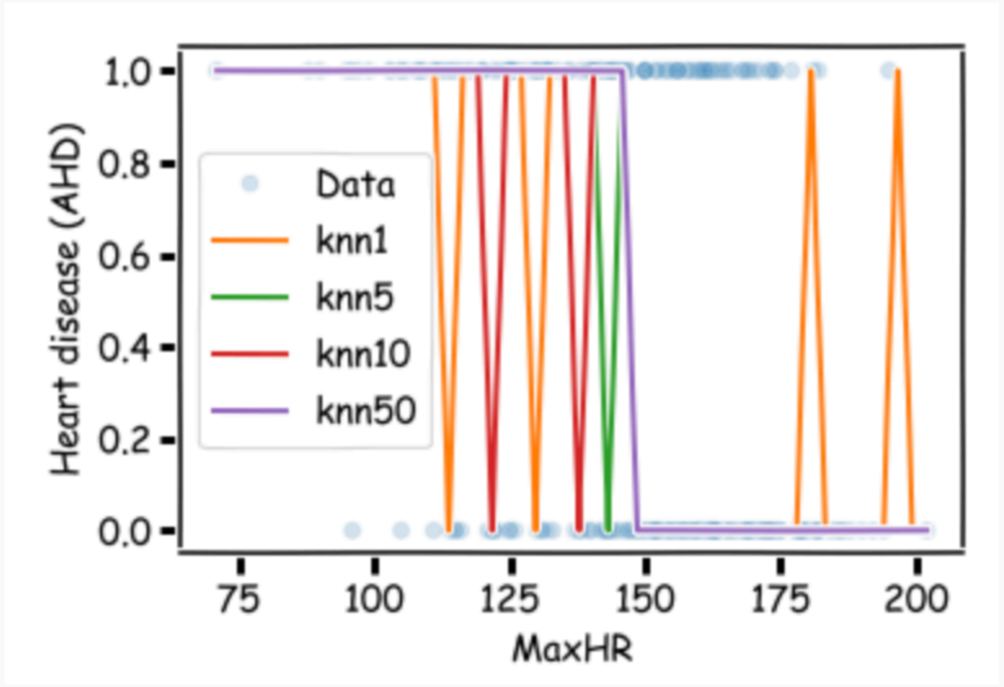
```
0.960396039604
0.712871287129
0.716171617162
0.706270627063
```

## Classification Boundaries in *k*-NN Classification

What will the classification boundaries look line in *k*-NN classification? With one predictors? With 2 predictors? With 3+ predictors?

How can we visualize these? In 1D? In 2D? In 3D+?

How do they compare to Logistic Regression classification boundaries?



## Dealing with Missingness

## What is missing data? k-Nearest Neighbors

Often times when data is collected, there are some missing values apparent in the dataset.  This leads to a few questions to consider:

- How does this show up in pandas?
- How does statsmodels and sklearn handle these NaNs?
- How does this effect our modeling?

## Naively handling missingness

What are the simplest ways to handle missing data?

- Drop the observations that have any missing values.
  - Use pd.DataFrame.dropna(axis=0)
- Impute the mean/median (if quantitative) or most common class (if categorical) for all missing values.
  - Use pd.DataFrame.fillna(value=x.mean())

How does statsmodels and sklearn handle these NaNs?

What are some consequences in handling missingness in this fashion?

## Missingness Indicator Variable

One simple way to handle missingness in a variable, $X_j$, is to impute a value (like 0 or $\bar{X}_j$), then create a new variable, $X_{j,miss}$, that indicates this observation had a missing value. If $X_j$ is categorical then just impute 0.

Then include both $X_{j,miss}$ and $X_j$ as predictors in any model.

Illustration is to the right.

| $X_1$ | $X_2$ | $X_1^*$ | $X_2^*$ | $X_{1,miss}$ | $X_{2,miss}$ |
|---|---|---|---|---|---|
| 10 | . | 10 | 0 | 0 | 1 |
| 5 | 1 | 5 | 1 | 0 | 0 |
| 21 | 0 | 21 | 0 | 0 | 0 |
| 15 | 0 | 15 | 0 | 0 | 0 |
| 16 | . | 16 | 0 | 0 | 1 |
| . | . | 0 | 0 | 1 | 1 |
| 21 | 1 | 21 | 1 | 0 | 0 |
| 12 | 0 | 12 | 0 | 0 | 0 |
| . | 1 | 0 | 1 | 1 | 0 |

## Why use a Missingness Indicator Variable?

How does this missingness indicator variable improve the model?

Because the group of individuals with a missing entry may be systematically different than those with that variable measured. Treating them equivalently could lead to bias in quantifying relationships (the $\beta$'s) and underperform in prediction.

For example: imagine a survey questions asks whether or not someone has ever recreationally used opioids, and some people chose not to respond. Does the fact that they did not respond provide extra information? Should we treat them equivalently as never-users?

This approach essentially creates a third group for this predictor: the "did not respond" group.

## Types of Missingness

## Sources of Missingness

Missing data can arise from various places in data:

- A survey was conducted and values were just randomly missed when being entered in the computer.
- A respondent chooses not to respond to a question like `Have you ever done cocaine?'.
- You decide to start collecting a new variable (like Mac vs. PC) partway through the data collection of a study.
- You want to measure the speed of meteors, and some observations are just 'too quick' to be measured properly.

The source of missing values in data can lead to the major types of missingness:

## Types of Missingness

There are 3 major types of missingness to be concerned about:

1. **Missing Completely at Random (MCAR)** - the probability of missingness in a variable is the same for all units. Like randomly poking holes in a data set.
2. **Missing at Random (MAR)** - the probability of missingness in a variable depends only on available information (in other predictors).
3. **Missing Not at Random (MNAR)** - the probability of missingness depends on information that has not been recorded and this information also predicts the missing values.

What are examples of each these 3 types?

## Missing completely at random (MCAR)

Missing Completely at Random is the best case scenario, and the easiest to handle:

- Examples: a coin is flipped to determine whether an entry is removed. Or when values were just randomly missed when being entered in the computer.
- Effect if you ignore: there is no effect on inferences (estimates of beta).
- How to handle: lots of options, but best to impute (more on next slide).

## Missing  at random (MAR)

Missing at random is still a case that can be handled.

- Example(s): men and women respond to the question "have you ever felt harassed at work?" at different rates (and may be harassed at different rates).
- Effect if you ignore: inferences are biased (estimates of $\beta$'s) and predictions are usually worsened.
- How to handle: use the information in the other predictors to build a model and **impute** a value for the missing entry.

Key: we can fix any biases by modeling and imputing the missing values based on what is observed!

# Missing Not at Random (MNAR)

Missing Not at Random is the worst case scenario, and impossible to handle properly:

- Example(s): patients drop out of a study because they experience some really bad side effect that was not measured. Or cheaters are less likely to respond when asked if you've ever cheated.
- Effect if you ignore: there is no effect on inferences (estimates of beta) or predictions.
- How to handle: you can 'improve' things by dealing with it like it is MAR, but you [likely] may never completely fix the bias. And incorporating a **missingness indicator variable** may actually be the best approach (if it is in a predictor).

## What type of missingness is present?

Can you ever tell based on your data what type of missingness is actually present?

Since we asked the question, the answer must be no.

It generally cannot be determined whether data really are missing at random, or whether the missingness depends on unobserved predictors or the missing data themselves. The problem is that these potential “lurking variables” are unobserved (by definition) and so can never be completely ruled out.

In practice, a model with as many predictors as possible is used so that the `missing at random' assumption is reasonable.

## Imputation Methods

## Handling Missing Data

When encountering missing data, the approach to handling it depends on:

1. whether the missing values are in the response or in the predictors.  Generally speaking, it is much easier to handle missingness in predictors.
2. whether the variable is quantitative or categorical.
3. how much missingness is present in the variable.  If there is too much missingness, you may be doing more damage than good.

Generally speaking, it is a good idea to attempt to **impute** (or 'fill in') entries for missing values in a variable (assuming your method of imputation is a good one).

# Imputation Methods

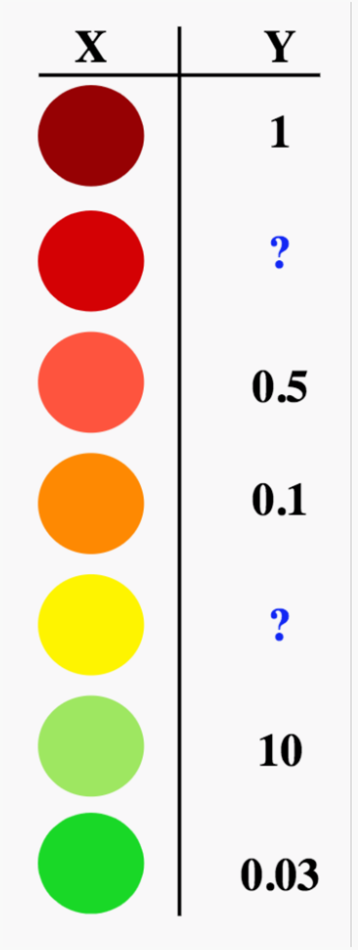
There are several different approaches to imputing missing values:

1. **Impute the mean or median** (quantitative) or most common class (categorical) for all missing values in a variable.
2. Create a new variable that is an **indicator of missingness**, and include it in any model to predict the response (also plug in zero or the mean in the actual variable).
3. **Hot deck imputation**: for each missing entry, randomly select an observed entry in the variable and plug it in.
4. **Model the imputation**: plug in predicted values ($\hat{y}$) from a model based on the other observed predictors.
5. **Model the imputation with uncertainty**: plug in predicted values plus randomness ($\hat{y} + \epsilon$) from a model based on the other observed predictors.

What are the advantages and disadvantages of each approach?

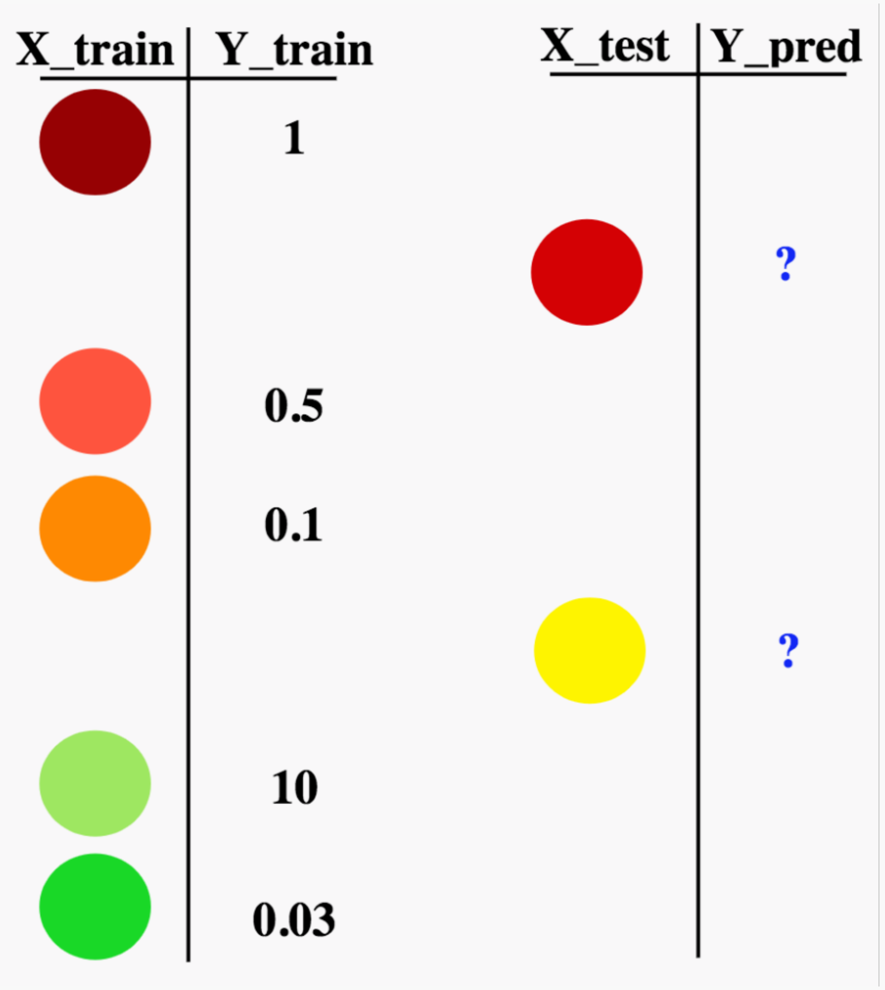
## Schematic: imputation through modeling

How do we use models to fill in missing data?



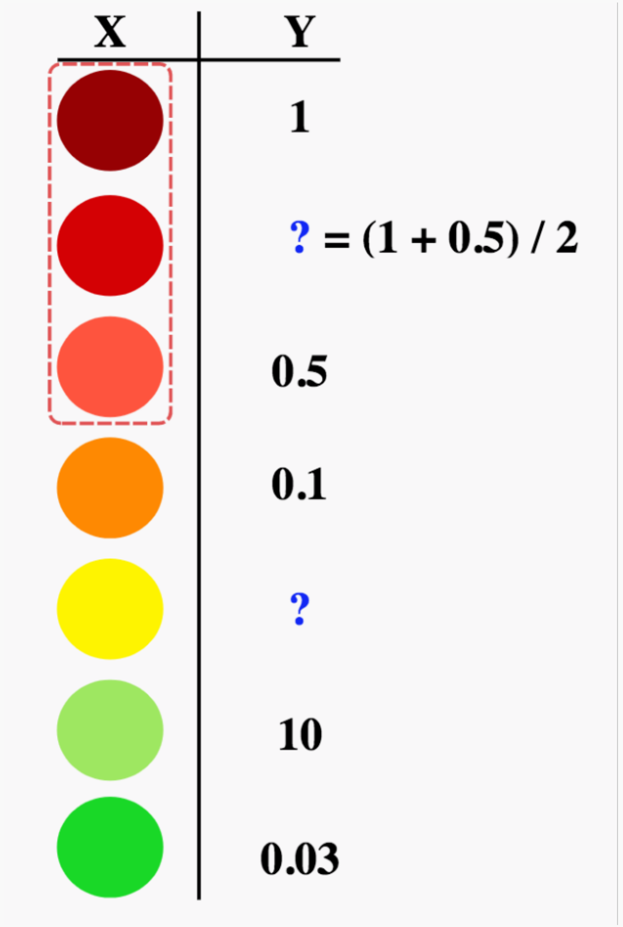
## Schematic: imputation through modeling

How do we use models to fill in missing data?



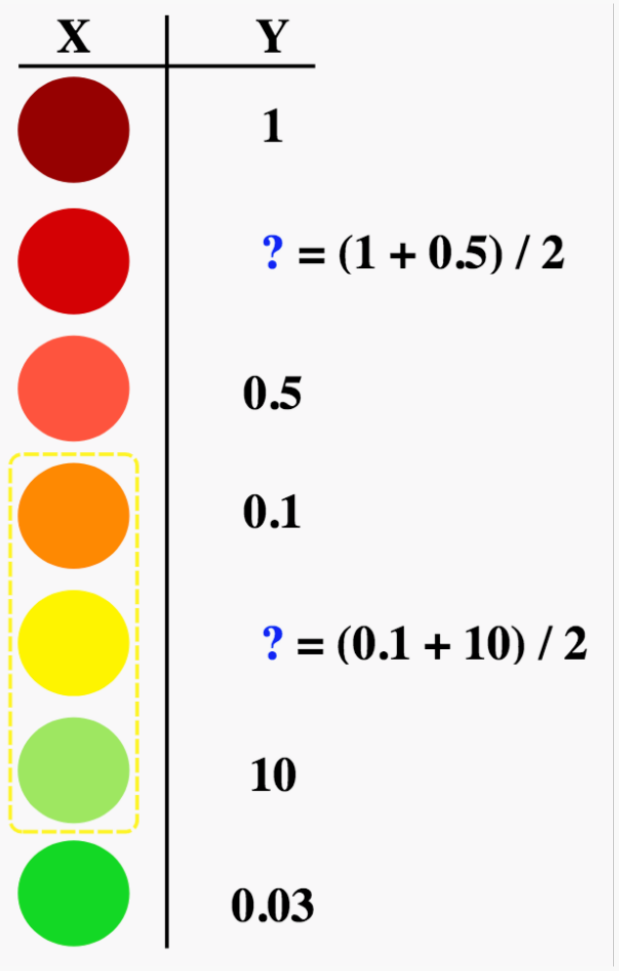
## Schematic: imputation through modeling

How do we use models to fill in missing data? Using *k*-NN for *k* = 2?



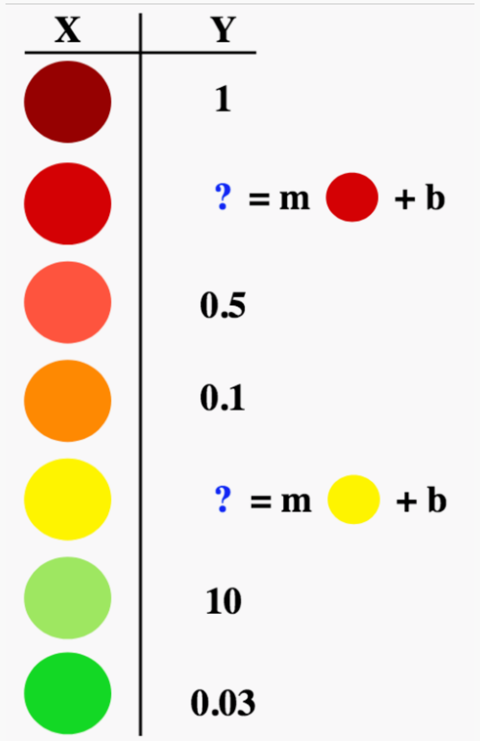
## Schematic: imputation through modeling

How do we use models to fill in missing data? Using *k*-NN for *k* = 2?



# Schematic: imputation through modeling

How do we use models to fill in missing data? Using linear regression?



Where *m* and *b* are computed from the observations (rows) that do not have missingness (we should call them *b* = $\beta_0$ and *m* = $\beta_1$).

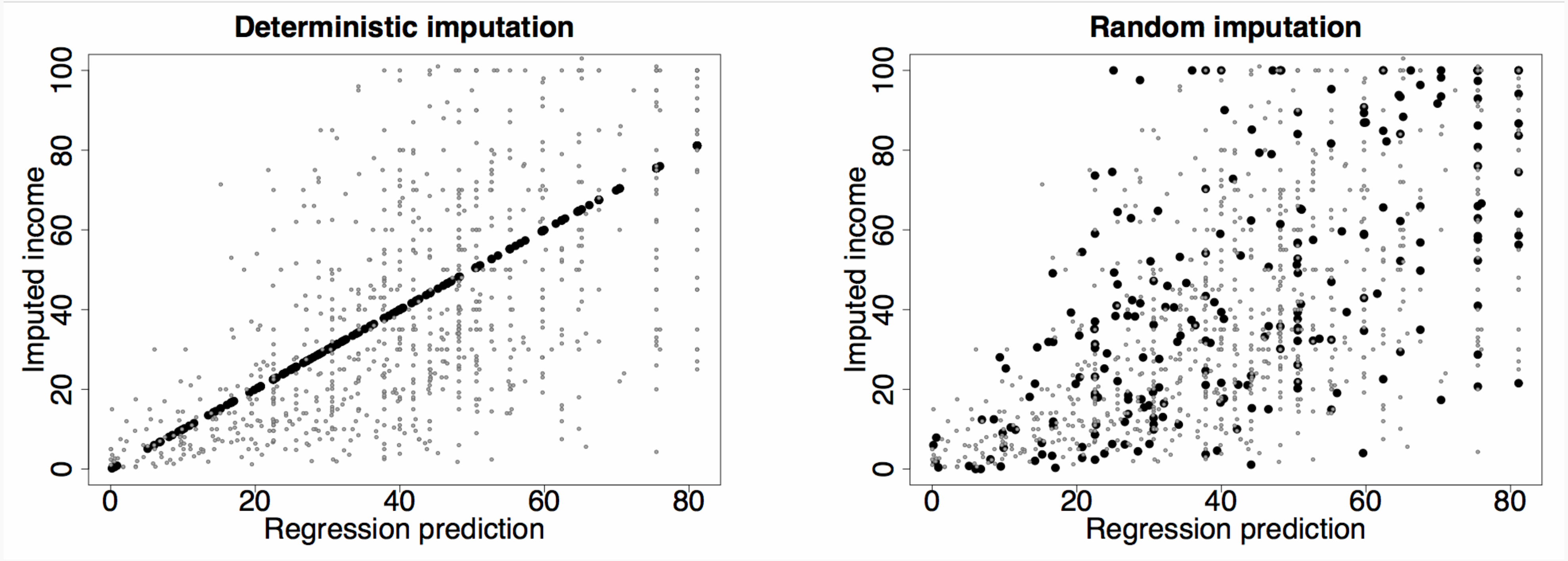
## Imputation through modeling with uncertainty

The schematic in the last few slides ignores the fact of imputing with uncertainty. What happens if you ignore this fact and just use the ‘best’ model to impute values solely on $\hat{y}$?

The distribution of the imputed values will be too narrow and not represent real data (see next slide for illustration). The goal is to impute values that include the uncertainty of the model.

How can this be done in practice in *k*-NN? In linear regression? In logistic regression?

# Imputation through modeling with uncertainty: an illustration



## Imputation through modeling with uncertainty: an illustration

Recall the probabilistic model in linear regression:

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \varepsilon$$

where $\epsilon \sim N(0, \sigma^2)$. How can we take advantage of this model to impute with uncertainty?

It's a 3 step process:

1. Fit a model to predict the predictor variable with missingness from all the other predictors.
2. Predict the missing values from the model in the previous part.
3. Add in a measure of uncertainty to this prediction by randomly sampling from a $N(0, \sigma^2)$ distribution, where $\sigma^2$ is the mean square error (MSE) from the model.

## Imputation through modeling with uncertainty: *k*-NN regression

How can we use *k*-NN regression to impute values that mimic the error in our observations?

Two ways:

- Use $k = 1$.
- Use any other *k*, but randomly select from the nearest neighbors in $\mathcal{N}_0$. This can be done with equal probability or with some weighting (inverse to the distance measure used).

## Imputation through modeling with uncertainty: classifiers

For classifiers, this imputation with uncertainty/randomness is a little easier process. How can it be implemented?

If a classification model (logistic, *k*-NN, etc...) is used to predict the variable with missingness on the observed predictors, then all you need to do is flip a ‘biased coin’ (or multi-sided die) with the probabilities of coming up for each class equal to the predicted probabilities from the model.

Warning: do not just classify blindly using the predict command in sklearn!

## Imputation across multiple variables

If only one variable has missing entries, life is easy. But what if all the predictor variables have a little bit of missingness (with some observations having multiple entries missing)? How can we handle that?

It’s an iterative process. Impute $X_1$ based on $X_2$, ..., $X_p$. Then impute $X_2$ based on $X_1$ and $X_3$, ..., $X_p$. And continue down the line.

Any issues? Yes, not all of the missing values may be imputed with just one ’run’ through the data set. So you will have to repeat these ’runs’ until you have a completely filled in data set.

## Multiple imputation: beyond this class

What is an issue with treating your now ‘complete’ data set (a mixture of actually observed values and imputed values) as simply all observed values?

Any inferences or predictions carried out will be tuned and potentially overfit to the random entries imputed for the missing entries. How can we prevent this phenomenon?

By performing **multiple imputation**: rerun the imputation algorithm many times, refit the model on the response many times (one time each), and then ’average’ the predictions or estimates of $\beta$ coefficients to perform inferences (also incorporating the uncertainty involved).

Note: this is beyond what we would expect in this class. But it generally a good thing to be aware of.