



< Previous						Next >
------------	--	--	--	--	--	--------

kNN for Regression

Bookmark this page



The k-Nearest Neighbors (kNN) algorithm works for both regression and classification. In both cases, a kNN model makes predictions on new data points using the k most similar points from our training dataset.

kNN for Regression

For each value of k, our output value is the average of the k nearest neighbors' outputs:

$$f(x) = \frac{1}{k} \{ f(y) \mid y \text{ is a } k\text{-nearest neighbor of } x \}$$

kNN for Classification

We classify a data point x_0 based on the labels of its neighbors

$$P(Y = j | X = x_0) = \frac{1}{k} \sum_{i \in N_0} I(y_i = j)$$

N_0 is defined as the set of the k nearest neighbors to x_0 .

I is the indicator function, equal to 1 when neighbor i has the label j and equal to 0 otherwise.

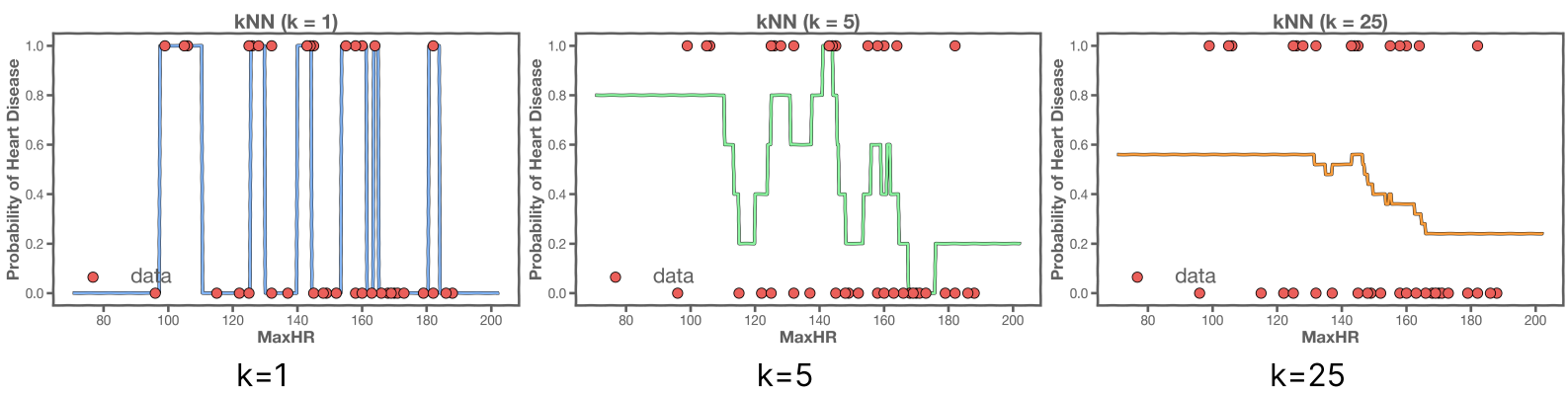
This formula builds a probability distribution for the class as the relative frequency of the classes in the set of neighbors N_0 .

For example, if there are $k = 50$ neighbors of x_0 , and 10 of the neighbors have label j , then the probability that x_0 has that same label j is given by

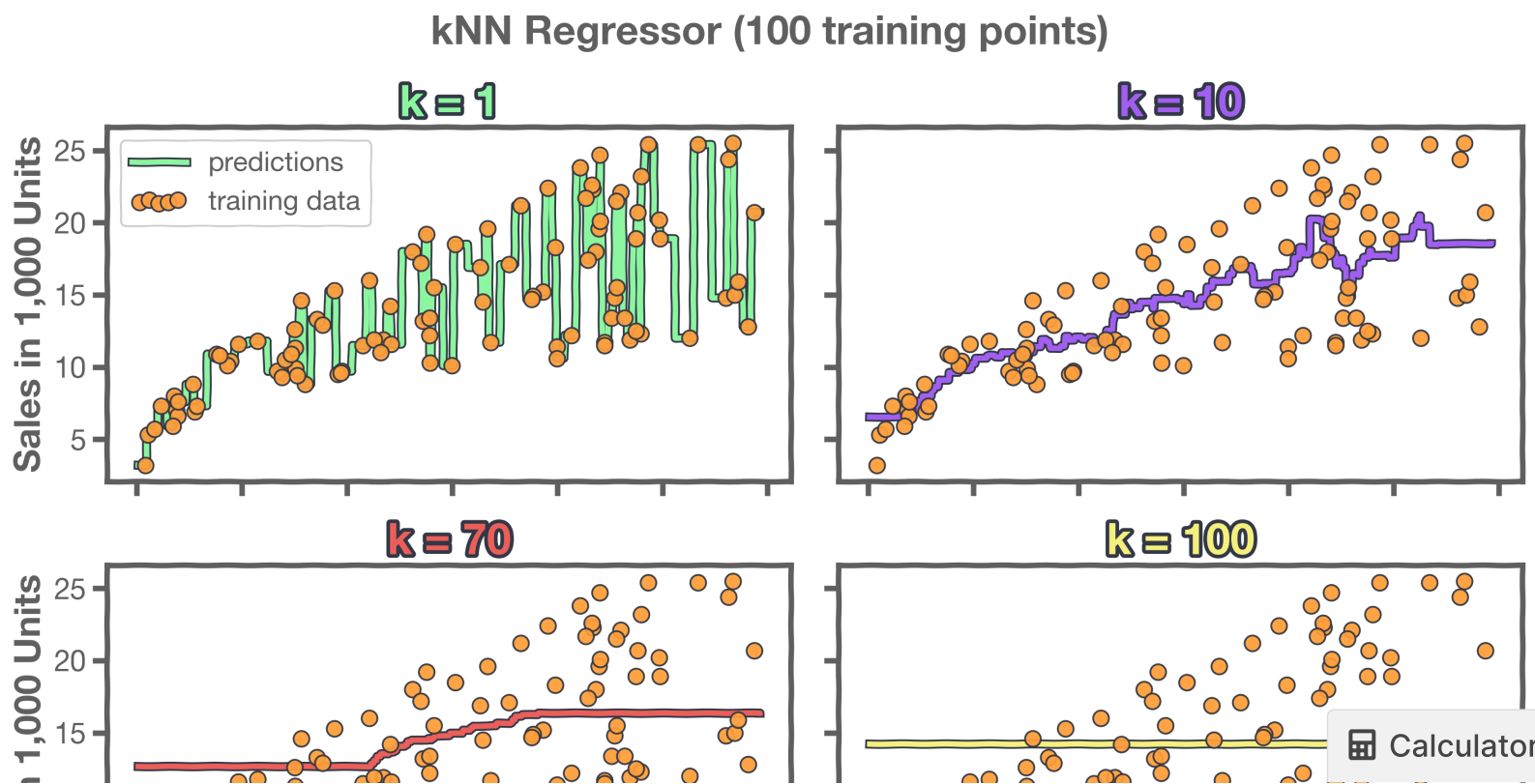
$$P(Y = j | X = x_0) = \frac{10}{50} = 0.2$$

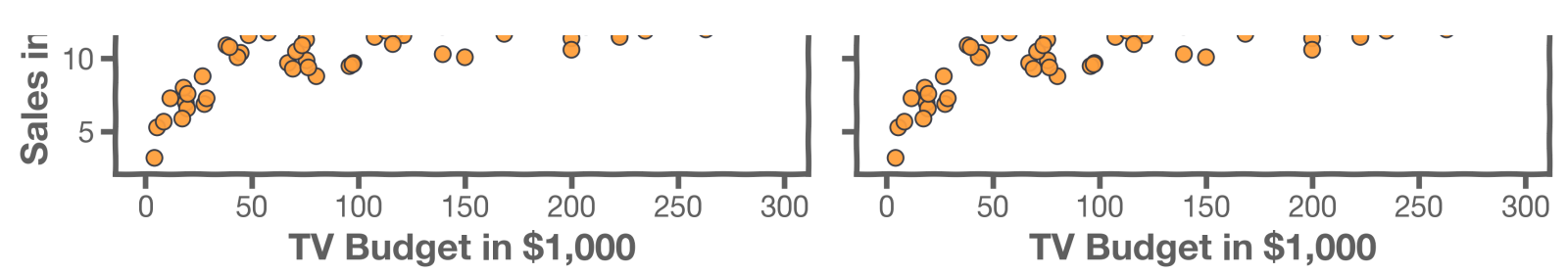
Changing the number of neighbors k: how smooth should our model be?

Notice that the behavior of kNN for classification is similar to kNN for regression: **as we increase k, we tend to see a smoother pattern in our model.** Here's an example:



Here's another set of examples:





How do we identify similar data points when the data has many features?

Here we walk through an example of **standardizing** a dataset so that we can get a better "distance" measurement between multi-dimensional data that includes a mix of quantitative and categorical features. In Python, you can use `sklearn.preprocessing.StandardScaler` to do this.

Suppose you are working for a movie ticket purchasing website. Users create an account for your website through an existing social media app. Your job is to build a model that can predict a user's favorite movie genre. You have access to some survey data your company recently collected:

feature	data type
year born	integer
new user	boolean, True for new users, False otherwise
# social media friends	integer
favorite genre	categorical from 4 options: Comedy, Action, Romance, and SciF

Using this data, you can build a k-nearest neighbors algorithm to try and predict, for other users not included in the dataset, what their favorite genre is.

Original unscaled data

[Skip to after table](#)

Year Born	New /Existing User	# Friends	Favorite Genre
1998	0	312	Action
1992	1	65	SciFi
2001	1	1923	Comedy
1987	0	203	Romance
1974	1	767	Romance
2000	0	54	Action

Consider a user born in 1990, who is an existing user with 1000 friends. We represent this new user as

User = [1990, 0, 1000]

and compute a Euclidean distance measurement with our existing users in our dataset. For example, the distance between User and our first row of data is given by

$$D([1990, 0, 1000], [1998, 0, 312]) = \sqrt{(1990 - 1998)^2 + (0 - 0)^2 + (1000 - 312)^2}$$

[Skip to after table](#)

Year Born	New/Existing User	# Friends	Favorite Genre	Euclidean Distance to User
1998	0	312	Action	688.05
1992	1	65	SciFi	935.00
2001	1	1923	Comedy	923.07
1987	0	203	Romance	797.00
1974	1	767	Romance	233.55
2000	0	54	Action	946.05

Now we can predict User's favorite genre using k=3 nearest neighbors. The nearest neighbors


 Calculator

1, 4, and 5. We assign a 2/3 probability that User's favorite genre is Romance based on the two neighbors who prefer Romance, and a 1/3 probability the favorite genre is Action based on the one neighbor who prefers Action- and therefore we would predict the label Romance.

But what is possibly going wrong here? Our distance measurement is flawed because most of the distance is coming from the difference in the # of friends, therefore under-reacting to the difference in year born. We have not given any reason why we should want out distance measurement to depend more on one feature than another. In the absence of such an explicit reason we should have the distance measure treat all features equally.

Scaled data using standard scaler

One way to get a balanced distance between data, regardless of the specific features, is to "standardize".

 STANDARDIZATION

Standardizing means applying this formula:

$$x_i \rightarrow \frac{x_i - mean(x_i)}{stdev(x_i)}$$

to each feature x_i for all the rows in the dataset.

Scaled User: [-0.213, -1.0, 0.679]

[Skip to after table](#)

Scaled Year Born	Scaled New/Existing User	Scaled # Friends	Favorite Genre	Euclidean Distance to Scaled User
0.638	-1.0	-0.368	Action	1.349
0.0	1.0	-0.744	SciFi	2.464
0.958	1.0	2.083	Comedy	2.710
-0.532	-1.0	-0.534	Romance	1.254
-1.915	1.0	0.324	Romance	2.650
0.851	-1.0	-0.761	Action	1.790

Now we can predict User's favorite genre using k=3 nearest neighbors. This time-nearest neighbors are rows 1, 4, and 6. We assign a 2/3 probability that User's favorite genre is Action based on the two neighbors who prefer Action, and a 1/3 probability that the favorite genre is Romance based on the one neighbor who prefers Romance- and therefore we would predict the label Action.

Is this a better distance function, and is it better that this time our model labeled Action instead of Romance? Well, without the actual labels for the data we cannot know for certain. But this time, with scaled data, our model is recognizing the similarities/differences in age at the same level of importance as the similarities/differences in the # of friends. So we can at least have more confidence that our model will be learning more directly from patterns in the data.

< Previous

Next >

[Affiliates](#)
[edX for Business](#)
[Open edX](#)
[Careers](#)
[News](#)

Legal

[Terms of Service & Honor Code](#)
[Privacy Policy](#)
[Accessibility Policy](#)
[Trademark Policy](#)
[Sitemap](#)
[Cookie Policy](#)
[Your Privacy Choices](#)

Connect

[Idea Hub](#)
[Contact Us](#)
[Help Center](#)
[Security](#)
[Media Kit](#)



© 2024 edX LLC. All rights reserved.
深圳市恒宇博科技有限公司 [粤ICP备17044299号-2](#)