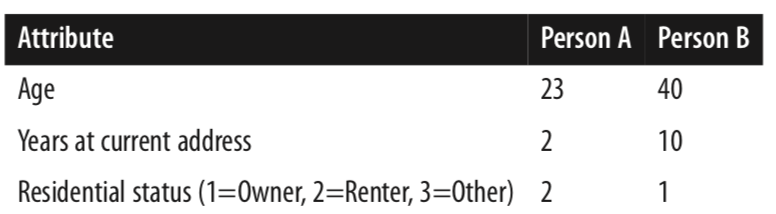
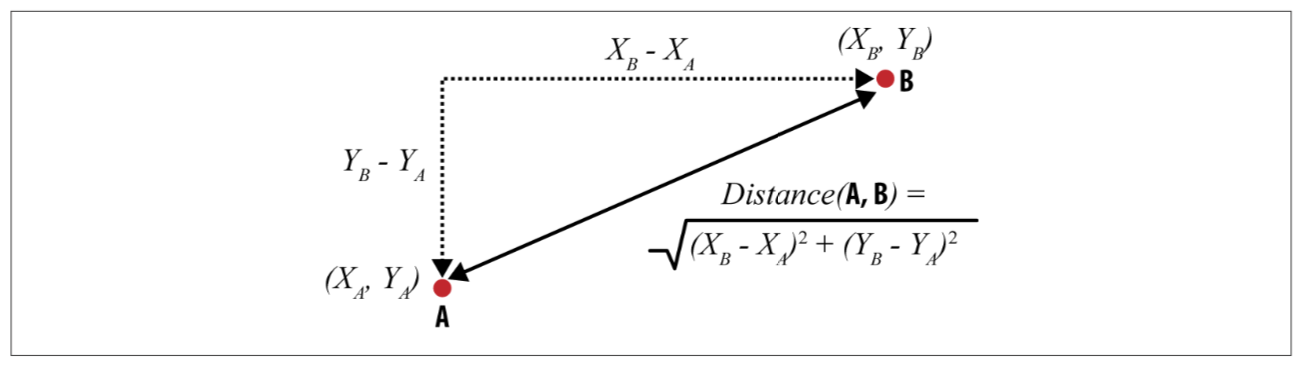
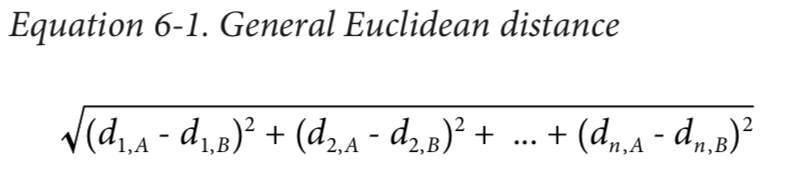
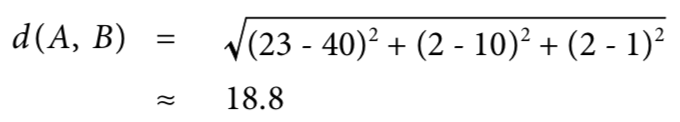
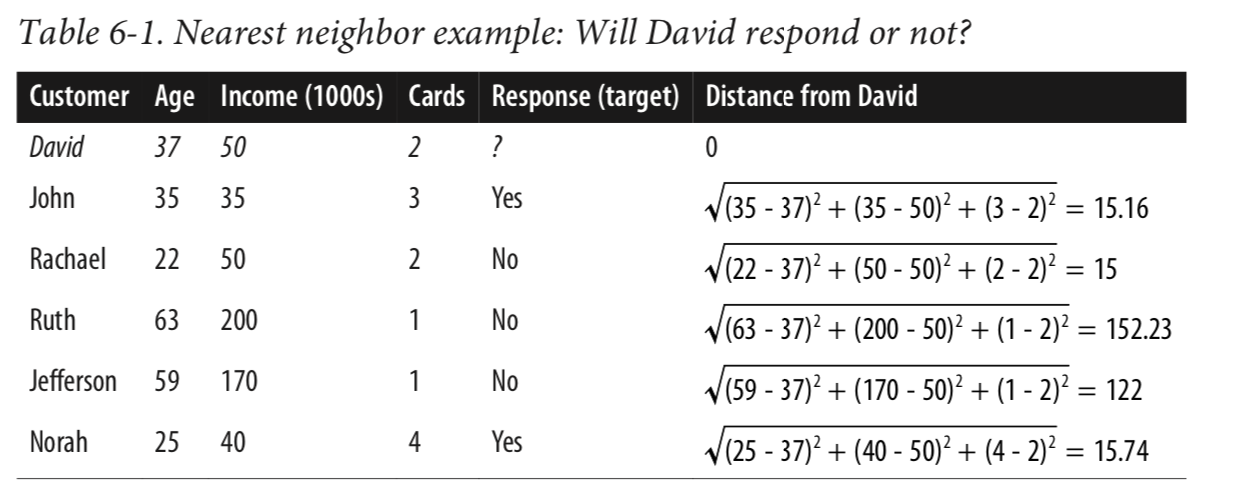
* Similarity can be used for doing *classification* and *regression*
* Modern retailers such as Amazon and Netflix use similarity to provide *recommendations* of similar products or from similar people. Whenever you see statements like “People who like X also like Y” or “Customers with your browsing history have also looked at ...” similarity is being applied.

Once an object can be represented as data, we can begin to talk more precisely about the similarity between objects, or alternatively the distance between objects. The closer two objects are in the space defined by the features, the more similar they are.



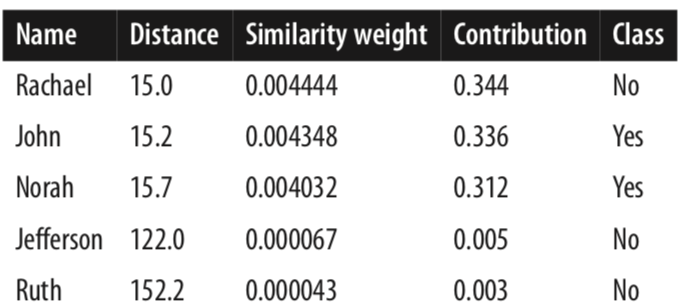




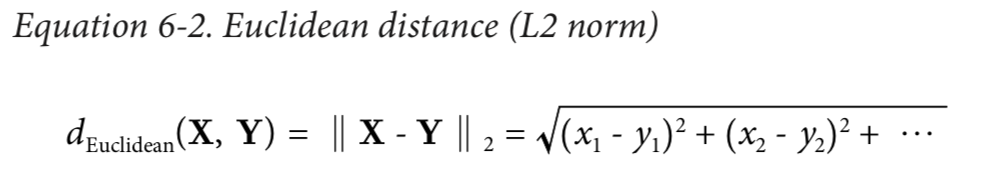


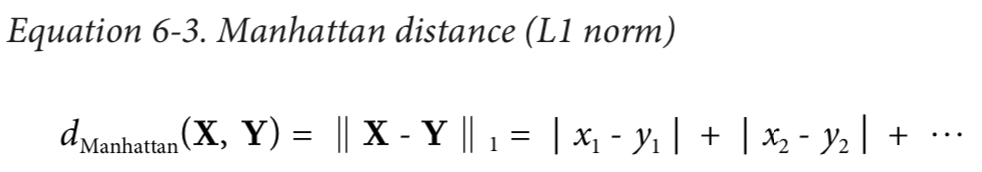
Three customers (John, Rachael, and Norah) are fairly similar to David, with a distance of about 15. The other two customers (Ruth and Jef‐ ferson) are much farther away. Therefore, David’s three **nearest neighbors** are Rachael, then John, then Norah. Their responses are No, Yes, and Yes, respectively. If we take a majority vote of these values, we predict Yes (David will respond).

**The same calculation can also be used for regression analysis** , say for eg want to predict David’s income based on the similarity with other people’s data.

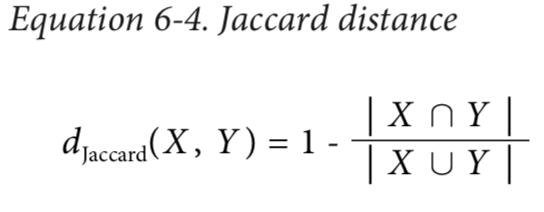


Think about what would happen if you evaluated a 1-NN classifier on the training data. When classifying each training data point, any reasonable distance metric would lead to the retrieval of that training point itself as its own nearest neighbor! Then its own value for the target variable would be used to predict itself, and voilà, perfect clas‐ sification. The same goes for regression. Hence 1-NN must overfit very strongly. In terms of overfitting and its avoidance, the *k* in a *k*-NN classifier is a complexity parameter. At one extreme, we can set *k* = *n* and we do not allow much complexity at all in our model.

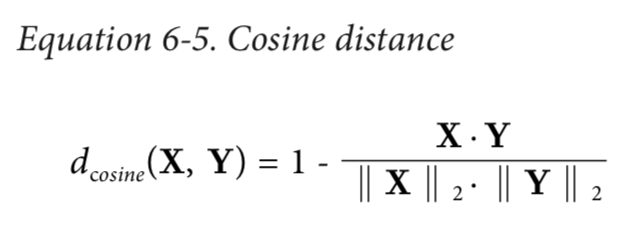
For setting other complexity parameters: we can conduct cross-validation or other nested holdout testing on the training set, for a variety of different values of *k*, searching for one that gives the best performance on the training data.



This simply sums the differences along the different dimensions between *X* and *Y*. It is called Manhattan (or taxicab) distance because it represents the total street distance you would have to travel in a place like midtown Manhattan to get between two points



Jaccard distance treats the two objects as *sets* of characteristics. Thinking about the objects as sets allows one to think about the size of the union of all the characteristics of two objects *X* and *Y*, |*X* ∪ *Y*|, and the size of the set of characteristics shared by the two objects (the in‐ tersection), |*X* ∩ *Y*|. Given two objects, *X* and *Y*, the Jaccard distance is the proportion of all the characteristics (that either has) that are shared by the two. This is appropriate for problems where the possession of a common characteristic between two items is important, but the common *absence* of a characteristic is not.



Cosine distance is particularly useful when you want to ignore differences in scale across instances—technically, when you want to ignore the magnitude of the vectors.