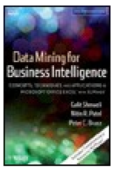

Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner, Second Edition

Chapter 7: *k*-Nearest Neighbors (*k*-NN)

In this chapter we describe the *k*-nearest neighbor algorithm that can be used for classification (of a categorical outcome) or prediction (of a numerical outcome). To classify or predict a new record, the method relies on finding "similar" records in the training data. These "neighbors" are then used to derive a classification or prediction for the new record by voting (for classification) or averaging (for prediction). We explain how similarity is determined, how the number of neighbors is chosen, and how a classification or prediction is computed. *k*-NN is a highly automated data-driven method. We discuss the advantages and weaknesses of the *k*-NN method in terms of performance and practical considerations such as computational time.

7.1 *k*-NN Classifier (Categorical Outcome)

The idea in *k*-nearest neighbor methods is to identify *k* records in the training dataset that are similar to a new record that we wish to classify. We then use these similar (neighboring) records to classify the new record into a class, assigning the new record to the predominant class among these neighbors. Denote by $(x_1, x_2, \ldots, x_p)$ the values of the predictors for this new record. We look for records in our training data that are similar or "near" the record to be classified in the predictor space (i.e., records that have values close to $x_1, x_2, \ldots, x_p$). Then, based on the classes to which those proximate records belong, we assign a class to the record that we want to classify.

Determining Neighbors

The fe-nearest neighbor algorithm is a classification method that does not make assumptions about the form of the relationship between the class membership (Y) and the predictors $X_1, X_2, \ldots, X_p$. This is a nonparametric method because it does not involve estimation of parameters in an assumed function form, such as the linear form assumed in linear regression (Chapter 6). Instead, this method draws information from similarities between the predictor values of the records in the dataset.

The central issue here is how to measure the distance between records based on their predictor values. The most popular measure of distance is the Euclidean distance. The Euclidean distance between two records $(x_1, x_2, \ldots, x_p)$ and $(u_1, u_2,$ $\ldots,$ *Up)is*

$$\sqrt{(x_1-u_1)^2+(x_2-u_2)^2+\cdots+(x_p-u_p)^2}. \tag{7.1}$$

You will find a host of other distance metrics in Chapters 12 and 14 for both numerical and categorical variables. However, the *k*-NN algorithm relies on many distance computations (between each record to be predicted and every record in the training set), and, therefore, the Euclidean distance, which is computationally cheap, is the most popular in *k*-NN.

To equalize the scales that the various predictors may have, note that in most cases predictors should first be standardized before computing a Euclidean distance.

Classification Rule

After computing the distances between the record to be classified and existing records, we need a rule to assign a class to the record to be classified, based on the classes of its neighbors. The simplest case is $k = 1$, where we look for the record that is closest (the nearest neighbor) and classify the new record as belonging to the same class as its closest neighbor. It is a remarkable fact that this simple, intuitive idea of using a single nearest neighbor to classify records can be very powerful when we have a large number of records in our training set. In turns out that the misclassification error of the one-nearest-neighbor scheme has a misclassification rate that is no more than twice the error when we know exactly the probability density functions for each class.

The idea of the **one-nearest neighbor** can be extended to $k > 1$ neighbors as follows:

1. Find the nearest *fe* neighbors to the record to be classified.
2. Use a majority decision rule to classify the record, where the record is classified as a member of the majority class of the *k* neighbors.

Example: Riding Mowers

A riding-mower manufacturer would like to find a way of classifying families in a city into those likely to purchase a riding mower and those not likely to buy one. A pilot random sample is undertaken of 12 owners and 12 nonowners in the city. The data are shown and plotted in Table 7.1. We first partition the data into training data (18 households) and validation data (6 households). Obviously, this dataset is too small for partitioning, which can result in unstable results, but we continue with this for illustration purposes. The training set is shown in Figure!7.1.

Table 7.1: LOT SIZE, INCOME, AND OWNERSHIP OF A RIDING MOWER FOR 24 HOUSEHOLDS

Household Number	Income($000s)	Lot Size (000s ft^2)	Ownership of Riding Mower
1	60.0	18.4	Owner
2	85.5	16.8	Owner
3	4.8	21.6	Owner
4	61.5	20.8	Owner
5	87.0	23.6	Owner
6	110.1	19.2	Owner
7	108.0	17.6	Owner
8	82.8	22.4	Owner
9	69.0	20.0	Owner
10	93.0	20.8	Owner
11	51.0	22.0	Owner
12	81.0	20.0	Owner
13	75.0	19.6	Nonowner
14	52.8	20.8	Nonowner
15	64.8	17.2	Nonowner
16	43.2	20.4	Nonowner
17	84.0	17.6	Nonowner
18	49.2	17.6	Nonowner
19	59.4	16.0	Nonowner
20	66.0	18.4	Nonowner
21	47.4	16.4	Nonowner
22	33.0	18.8	Nonowner
23	51.0	14.0	Nonowner
24	63.0	14.8	Nonowner

Now consider a new household with \$60, 000 income and lot size 20, 000 ft^2 (also shown in Figure 7.1). Among the households in the training set, the one closest to the new household (in Euclidean distance after normalizing income and lot size) is household 4, with \$61, 500 income and lot size 20, 800 ft^2. If we use a 1NN classifier, we would classify the new household as an owner, like household 4. If we use k = 3, the three nearest households are 4, 9, and 14. The first two are owners of riding mowers, and the last is a nonowner. The majority vote is therefore *owner*, and the new household would be classified as an owner.

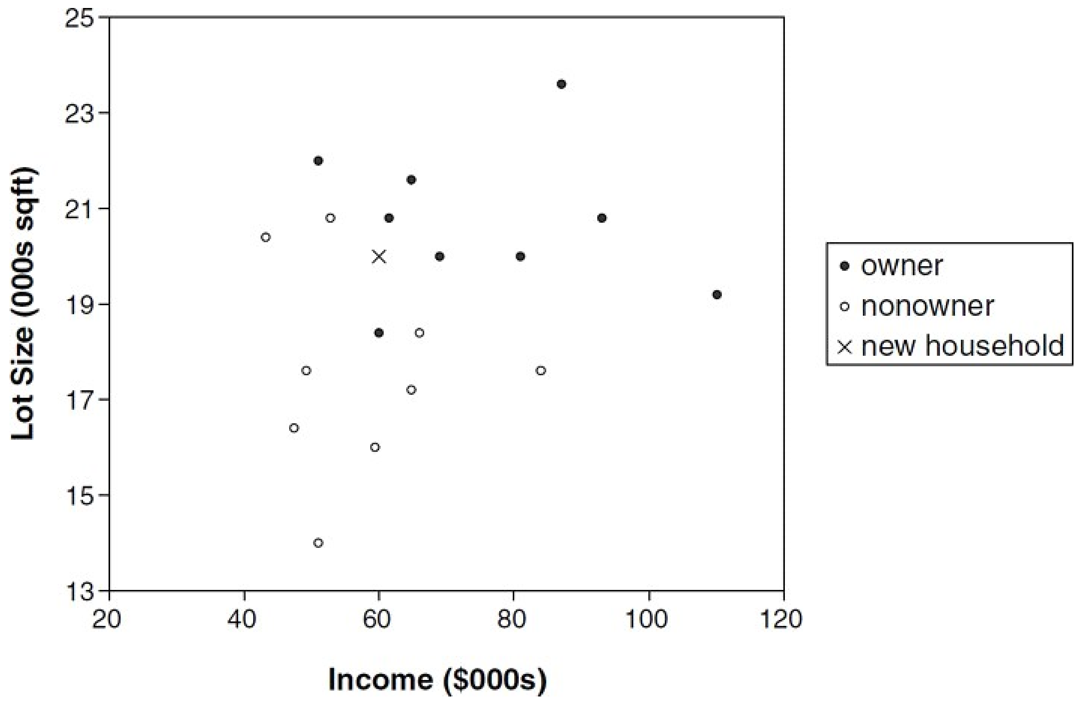

FIGURE 7.1: SCATTERPLOT OF LOT SIZE VS. INCOME FOR THE 18 HOUSEHOLDS IN THE TRAINING SET AND THE NEW HOUSEHOLD TO BE CLASSIFIED

Choosing *k*

The advantage of choosing *k* > 1 is that higher values of *k* provide smoothing that reduces the risk of overfitting due to noise in the training data. Generally speaking, if *k* is too low, we may be fitting to the noise in the data. However, if *k* is too high, we will miss out on the method's ability to capture the local structure in the data, one of its main advantages. In the extreme, *k* = *n* = the number of records in the training dataset. In that case we simply assign all records to the majority class in the training data, irrespective of the values of $(x_1, x_2, \ldots, x_p)$, which coincides with the naive rule! This is clearly a case of oversmoothing in the absence of useful information in the predictors about the class membership. In other words, we want to balance between overfitting to the predictor information and ignoring this information completely. A balanced choice depends greatly on the nature of the data. The more complex and irregular the structure of the data, the lower the optimum value of k. Typically, values of *k* fall in the range of 1-20. Often, an odd number is chosen to avoid ties.

So how is *k* chosen? Answer: We choose the *k* that has the best classification performance. We use the training data to classify the records in the validation data, then compute error rates for various choices of k. For our example, if we choose *k* = 1, we will classify in a way that is very sensitive to the local characteristics of the training data. On the other hand, ifwe choose a large value of k, such as *k* = 18, we would simply predict the most frequent class in the dataset in all cases. This is a very stable prediction, but it completely ignores the information in the predictors. To find a balance, we examine the misclassification rate (of the validation set) that results for different choices of *e* between 1 and 18. This is shown in Figure 7.2. We would choose *fe* = 8, which minimizes the misclassification rate in the validation set.[1] Note, however, that now the validation set is used as an addition to the training set and does not reflect a holdout set as before. Ideally, we would want a third test set to evaluate the performance of the method on data that it did not see.

Value of *k*	% Error Training	% Error Validation	
1	0.00	33.33	
2	16.67	33.33	
3	11.11	33.33	
4	22.22	33.33	
5	11.11	33.33	
6	27.78	33.33	
7	22.22	33.33	
8	22.22	16.67	<--- Best k
9	22.22	16.67	
10	22.22	16.67	
11	16.67	33.33	
12	16.67	16.67	
13	11.11	33.33	
14	11.11	16.67	
15	5.56	33.33	
16	16.67	33.33	
17	11.11	33.33	
18	50.00	50.00	

FIGURE 7.2: MISCLASSIFICATION RATE OF VALIDATION SET FOR VARIOUS CHOICES OF *K*

Once *k* is chosen, the algorithm uses it to generate classifications of new records. An example is shown in Figure 7.3, where eight neighbors are used to classify the new household.

Data range	['KNN Riding Mowers.xlsx']'Data_Partition1'!$G$20:$H$20

Cut off Prob.Val. for Success (Updatable)	**0.5**	(Updating the value here will NOT update)

Row Id.	Predicted Class	Prob. for Owner	Actual #Nearest Neighbors	Income	Lot Size
1	owner	0.625	8	60	20

FIGURE 7.3: CLASSIFYING A NEW HOUSEHOLD USING THE "BEST *K*" = 8

Setting the Cutoff Value

k-NN uses a majority decision rule to classify a new record, where the record is classified as a member of the majority class of the *fe* neighbors. The definition of "majority" is directly linked to the notion of a cutoff value applied to the class membership probabilities. Let us consider a binary outcome case. For a new record, the proportion of class 1 members among its neighbors is an estimate of its probability of belonging to class 1. In the riding-mower example with *k* = 3, we found that the three nearest neighbors to the new household (with income = $60, 000 and lot size = 20, 000 ft^2) are households 4, 9, and 14. Since 4 and 9 are owners and 14 is a nonowner, we can estimate for the new household a probability of 2/3 of being an owner (and 1/3 for being a nonowner). Using a simple majority rule is equivalent to setting the cutoff value to 0.5. Another example can be seen in Figure 7.3, where *k* = 8 was used to classify the new household. The "Prob for Owner" of 0.625 was obtained because 5 of the 8 neighbors were owners. Using a cutoff of 0.5 leads to a classification of "owner" for the new household.

As mentioned in Chapter 5, changing the cutoff value affects the classification matrix (i.e., the error rates). Hence, in some cases we might want to choose a cutoff other than the default 0.5 for the purpose of maximizing accuracy or for incorporating misclassification costs. In XLMiner this can be done by directly changing the cutoff value (as can be seen in Figure 7.3), which automatically changes the "Predicted Class" and the related classification matrices.

k-NN with More than Two Classes

The *k*-NN classifier can easily be applied to an outcome with *m* classes, where *m* > 2. The "majority rule" means that a new record is classified as a member of the majority class of its *k* neighbors. An alternative, when there is a specific class that we are interested in identifying (and are willing to "overidentify" records as belonging to this class), is to calculate the proportion of the *k* neighbors that belong to this class of interest, use that as an estimate of the probability that the new record belongs to that class, and then refer to a user-specified cutoff value to decide whether to assign the new record to that class. For more on the use of a cutoff value in classification where there is a single class of interest, see Chapter 5.

[1]Partitioning such a small dataset is unwise in practice, as results will heavily rely on the particular partition. For instance, if you use a different partitioning, you might obtain a different "optimal" fe. We use this example for illustration only.

7.2 *k*-NN for a Numerical Response

The idea of *k*-NN can readily be extended to predicting a continuous value (as is our aim with multiple linear regression models). The first step of determining neighbors by computing distances remains unchanged. The second step, where a majority vote of the neighbors is used to determine class, is modified such that we take the average response value of the fe-nearest neighbors to determine the prediction. Often, this average is a weighted average, with the weight decreasing with increasing distance from the point at which the prediction is required.

Another modification is in the error metric use for determining the "best fe." Rather than the overall error rate used in classification, RMSE (or another prediction error metric) is used in prediction (see Chapter 5).

PANDORA

Pandora is an Internet music radio service that allows users to build customized "stations" that play music similar to a song or artist that they have specified. Pandora uses a *k-NN* style clustering/classification process called the Music Genome Project to locate new songs or artists that are close to the user-specified songorartist. In simplified terms, the process works roughly as follows forsongs:

1. Pandora has established hundreds of variables on which a song can be measured on a scale from 0 to 5. Four such variables from the beginning of the list are
   - Acid Rock Qualities
   - Accordion Playing
   - Acousti-Lectric Sonority
   - Acousti-Synthetic Sonority
2. Pandora pays musicians to analyze tens of thousands of songs and rate each songon each of these attributes. Each songwill then be represented by a row vector of values between 0 and 5, for example, for Led Zeppelin's Kashmir:

   Kashmir 4 0 3 3… (high on acid rock attributes, no accordion, etc.)

   This step represents a costly investment and lies at the heart of Pandora's value because these variables have been tested and selected because they accurately reflect the essence of a song and provide a basis for defining highly individualized preferences.
3. The online user specifies a song that he/she likes (the song must be in Pandora's database).
4. Pandora then calculates the statistical distance[2] between the user's song and the songs in its database. It selects a song that is close to the user-specified song and plays it.
5. The user then has the option of saying "I like this song," "I don't like this song," or saying nothing.
6. If "like" is chosen, the original song, plus the new song are merged into a two-song cluster[3] that is represented by a single vector, comprised means of the variables in the original two-song vectors.
7. If "don't like" is chosen, the vector of the song that is not liked is stored for future reference. (If the user does not express an opinion about the song, in our simplified example here the new song is not used for further comparisons.)

8. Pandora looks in its database for a new song, one whose statistical distance is close to the "like" song cluster[4] and not too close to the "don't like" song. Depending on the user's reaction, this new song might be added to the "like" cluster or "don't like" cluster.

Over time, Pandora develops the ability to deliver songs that match a particular taste of a particular user. A single user might build up multiple stations around different song clusters. Clearly, this is a less limiting approach than selecting music in terms of which "genre" it belongs to (which is a more limitingapproach).

While the process described above is a bit more complex than the basic "classification of new data" process described in this chapter, the fundamental process—classifying a record according to its proximity to other records—is the same at its core. Note the role of domain knowledge in this machine learning process—the variables have been tested and selected by the project leaders, and the measurements have been made by human experts.

Further Reading: See www.pandora.com, Wikipedia's article on the Music Genome Project, and Joyce John's article "Pandora and the Music Genome Project," in the September 2006 *Scientific Computing* [23(10): 14, 40-41].

[2]See Chapter 12 for an explanation of statistical distance.

[3]See Chapter 14 for more on clusters.

[4]See Case 18.4 "Segmenting Consumers of Bath Soap" for an exercise involving the identification of clusters, which are then used for classification purposes.

7.3 Advantages and Shortcomings of *k*-NN Algorithms

The main advantage of *e*-NN methods is their simplicity and lack of parametric assumptions. In the presence of a large enough training set, these methods perform surprisingly well, especially when each class is characterized by multiple combinations of predictor values. For instance, in real estate databases there are likely to be multiple combinations of {home type, number of rooms, neighborhood, asking price, etc.} that characterize homes that sell quickly versus those that remain for a long period on the market.

There are two difficulties with the practical exploitation of the power of the *e*-NN approach. First, although no time is required to estimate parameters from the training data (as would be the case for parametric models such as regression), the time to find the nearest neighbors in a large training set can be prohibitive. A number of ideas have been implemented to overcome this difficulty. The main ideas are:

- Reduce the time taken to compute distances by working in a reduced dimension using dimension reduction techniques such as principal components analysis (Chapter 4).
- Use sophisticated data structures such as search trees to speed up identification of the nearest neighbor. This approach often settles for an "almost nearest" neighbor to improve speed.
- Edit the training data to remove redundant or almost redundant points to speed up the search for the nearest neighbor. An example is to remove records in the training set that have no effect on the classification because they are surrounded by records that all belong to the same class.

Second, the number of records required in the training set to qualify as large increases exponentially with the number of predictors *p*. This is because the expected distance to the nearest neighbor goes up dramatically with *p* unless the size of the training set increases exponentially with p. This phenomenon is known as the *curse of dimensionality*, a fundamental issue pertinent to all classification, prediction, and clustering techniques. This is why we often seek to reduce the number of predictors through methods such as selecting subsets of the predictors for our model or by combining them using methods such as principal components analysis, singular value decomposition, and factor analysis (see Chapter 4).

Problems

7.1 **Personal Loan Acceptance**. Universal Bank is a relatively young bank growing rapidly in terms of overall customer acquisition. The majority of these customers are liability customers (depositors) with varying sizes of relationship with the bank. The customer base of asset customers (borrowers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business. In particular, it wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors).

A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise smarter campaigns with better target marketing. The goal of our analysis is to model the previous campaign's customer behavior to analyze what combination of factors make a customer more likely to accept a personal loan. This will serve as the basis for the design of a new campaign.

The file UniversalBank.xls contains data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign *(Personal Loan)*. Among these 5000 customers, only 480(= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

Partition the data into training (60%) and validation (40%) sets.

a. Perform a *k*-NN classification with all predictors except ID and ZIP code using *fe* = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the *success* class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified?

b. What is a choice of *k* that balances between overfitting and ignoring the predictor information?

c. Show the classification matrix for the validation data that results from using the best fe.

d. Classify the customer using the best *e*.

e. Repartition the data, this time into training, validation, and test sets (50%: 30%: 20%). Apply the fe-NN method with the *fe* chosen above. Compare the classification matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

7.2 **Predicting Housing Median Prices**. The file BostonHousing.xls contains information on over 500 census tracts in Boston, where for each tract 14 variables are recorded. The last column (CAT.MEDV) was derived from MEDV such that it obtains the value 1 if MEDV> 30 and 0 otherwise. Consider the goal of predicting the median value (MEDV) of a tract, given the information in the first 13 columns.

Partition the data into training (60%) and validation (40%) sets.

a. Perform a *k*-NN prediction with all 13 predictors (ignore the CAT.MEDV column), trying values of *k* from 1 to 5. Make sure to normalize the data (click "normalize input data"). What is the best *k* chosen? What does it mean?

b. Predict the MEDV for a tract with the following information, using the best *e:*

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD
0.2	0	7	0	0.538	6	62	4.7	4
TAX	PTRATIO	B	LSTAT					
307	21	360	10					

(Copy this table with the column names to a new worksheet and then in "Score new data" choose "from worksheet")

c. Why is the error of the training data zero?

d. Why is the validation data error overly optimistic compared to the error rate when applying this *k*-NN predictor to new data?

e. If the purpose is to predict MEDV for several thousands of new tracts, what would be the disadvantage of using *k*-NN prediction? List the operations that the algorithm goes through in order to produce each prediction.