Big Data and Better Decisions
Spring 2018

Module 9: Nearest Neighbor Matching

1. The Classification Setting

In this half of the course, we'll focus on situations where we want to make **predictions** about a particular outcome using detailed data. This is in contrast to the first half of the course, where we focused on using data to make **inferences** about what causes an outcome (and in some cases, how we might change it). Sometimes, the outcome that we are interested in predicting is a continuous variable like the price of a consumer good. Here, we'll focus instead on prediction when our outcome of interest is a category, classification or binary variable. For example, this could be which of 5 brands of cereal a consumer chooses to buy, or whether a drug trial succeeds or fails. In these notes we will focus on the simple case of a binary outcome, or an outcome with only two categories: Success or Failure. For the more general case, see section 2.2.3 of Introduction to Statistical Learning (ISL).

To develop our prediction model, we'll need a **training** dataset and a **test** dataset. We fit the model using our training dataset, and then test its fit on the test dataset. For example, if we were trying to predict whether a consumer was likely to purchase a particular product, we might have a dataset of past purchasing behavior linked to individual consumer demographics. If that dataset had 5000 observations, we could use 4000 as a training dataset and leave 1000 to test our model. We do this to prevent us from overfitting our model to the training dataset (past purchasing data), which leads it to perform poorly when we apply it to the actual outcome we want it to predict (future purchasing behavior). The **training error rate** tells us how well our model fits the training dataset. In the classification setting, where y is the true outcome and $\hat{y}$ is the outcome that our model predicts, the error rate is just the mean rate of unsuccessful prediction in the training dataset, or:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \widehat{y_i})$$

As mentioned above, we don't want the lowest possible error rate on the training dataset, because then our model will not perform well when we apply it to our real world situation. Instead, we care about the **test error rate**, which is calculated the same way as the training error rate but using the actual outcomes in the test dataset and the predicted outcomes based on covariates from the test dataset. So, we want to pick the model for which the test error rate is the smallest.

Consider a case where we are trying to predict the outcome Y equal to Success or Failure based on a predictor variable X. It can be shown that the test error rate is minimized when we assign each observation to Success when the conditional probability of success conditional on the predictor variable is greater than 0.5, or:

$$\Pr(Y = Success|X = x_0) > 0.5 \rightarrow \hat{Y} = Success$$

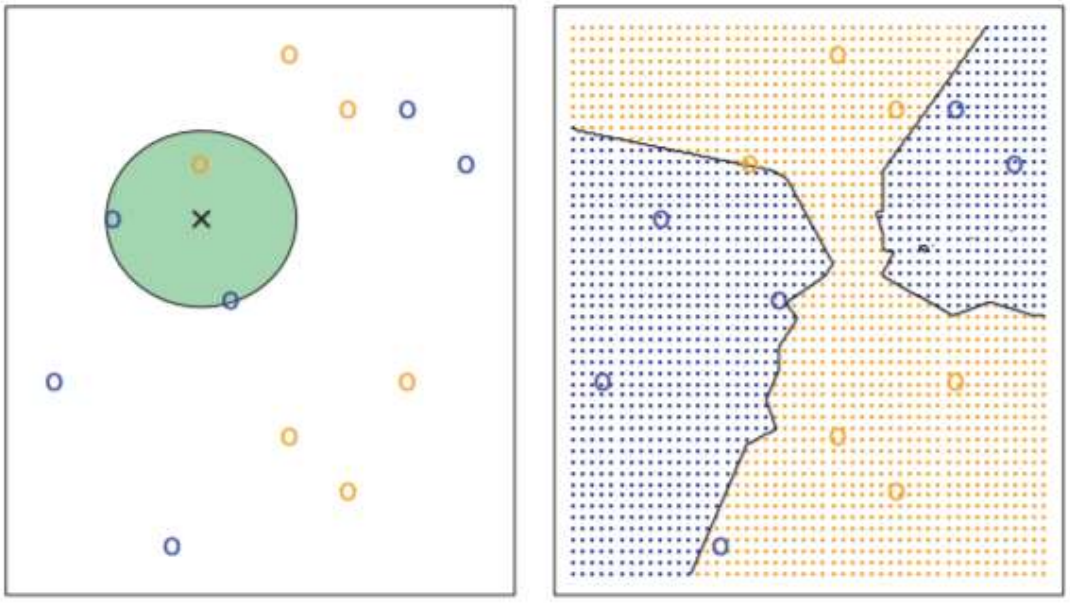
Think about expanding this to include a *vector* of predictor variables (more than one variable)—this is

called the **Bayes classifier**. In theory, we'd like to predict using the Bayes classifier, but we don't know the true underlying conditional probabilities for our real data. Instead, we can use approaches that estimate the conditional distribution of Y given X, and classify observations based on the highest *estimated* probability. One of these methods is the K-nearest neighbors (KNN) classifier.

2. K-Nearest Neighbors

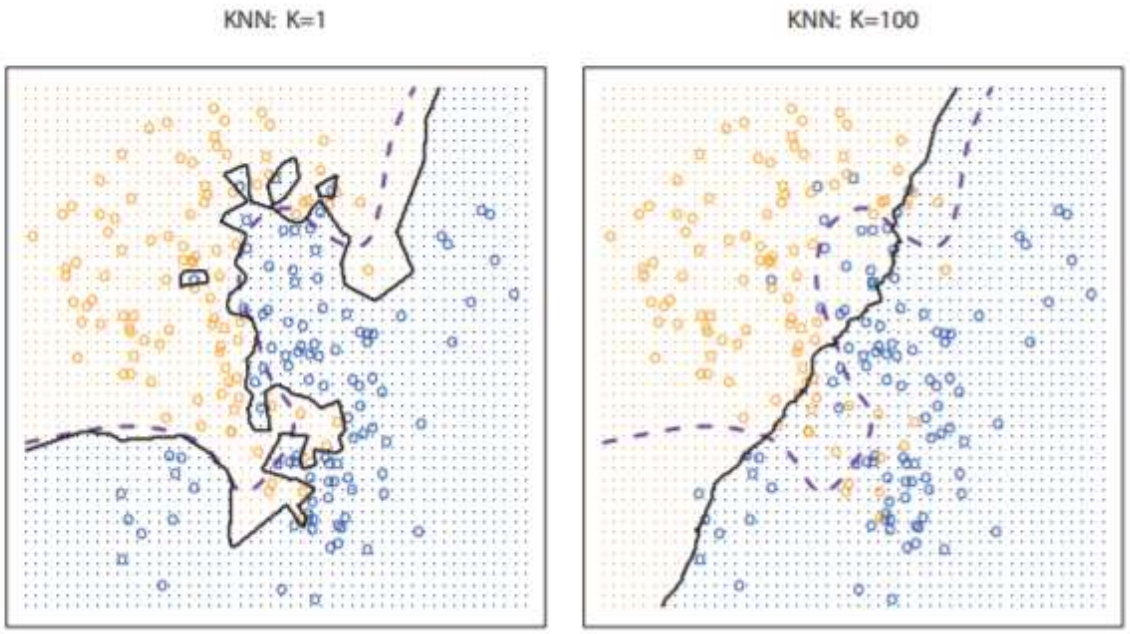
KNN works by taking an observation from the *test* dataset and identifying a number of observations K in the *training* dataset whose predictor variable values are closest to its own. It then estimates the conditional probability of Success by using the fraction of these surrounding observations whose actual outcome is Success. We can pick any number of observations K. To understand about how we identify which training observations are "closest", think about an example with two predictor variables. We could create a graph with one of the variables on the x-axis, and one on the y-axis, and plot all of the points from the training dataset and one point from the test dataset. We could identify which points are closest by drawing lines between the test point and each of the training points and selecting the training points with the shortest lines. With 3 predictor variables, we would be drawing our lines in a 3-dimensional cube. In practice, we'll be using many more predictor variables, so the notion of distance may be hard to visualize, but the same concept applies.

From ISL: see the figures below using K = 3, in a simple situation with six blue observations and six orange observations, and two predictor variables. Left: a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue. Right: The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.

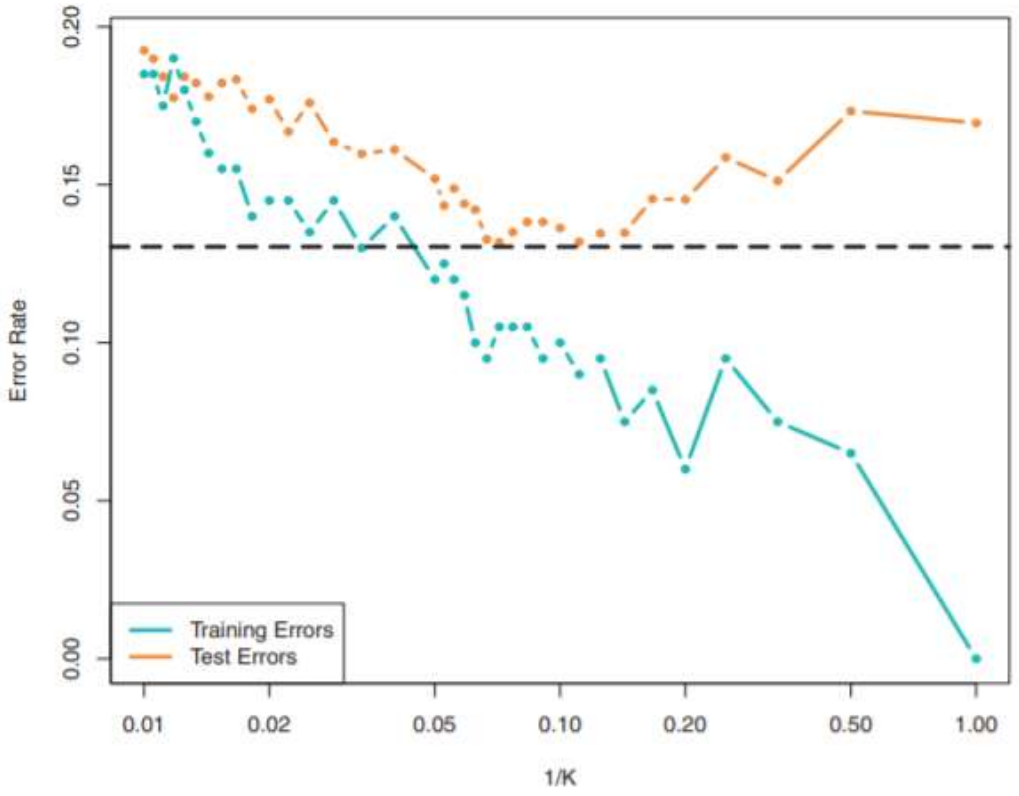


We said above that we can choose any value of K. The smaller the value of K, the more flexible our model, and the lower our training error rate will be. In fact, with K=1, the training error rate will be 0 (because the predicted success rate equals 1 if each observation in the training dataset succeeded and 0 otherwise). See an illustration below of how choice of K affects prediction success for a training dataset with many observations (you can see that prediction is perfect for K=1). The purple line

represents the decision boundary based on the actual underlying probabilities of blue vs orange (which are known in this case only because this is simulated data):



However, the same tradeoffs between variance and bias that we discussed in lecture apply in this case, so a lower training error rate does not necessarily mean that our testing error rate will be lower, which is what we ultimately care about. The graph below uses the same data as the one above and shows that though the training error rate is indeed minimized when n=1. However, the testing error rate is quite high when n=1, and is actually minimized roughly around K=10.



3. Implementation in R

We will practice an example of KNN reproduced from section 4.6.6 of ISL, which uses the Caravan auto insurance dataset (loadable from JupyterHub). This data set includes 85 predictors that measure demographic characteristics for 5,822 individuals. The response variable is Purchase, which indicates whether or not a given individual purchases an auto insurance policy. In this data set, only 6% of people purchased auto insurance. A data dictionary that elaborates on the variable names is listed in the Appendix.

First we will read the dataset AutoInsurance.csv into R. We will then use the dim and the summary functions to understand the structure and summary of the response variable.

```
# Examine the data
dim(Insurance_Data)
# you can attach a dataset so that you don't have to reference
it every time
attach(Insurance_Data)
summary(Purchase)
# Purchase Rate
348/5474
```

Because The KNN classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters. Any variables that are on a large scale will have a much larger effect on the distance between the observations, and hence on the KNN classifier, than variables that are on a small scale. For instance, imagine a data set that contains two variables, salary and age (measured in dollars and years, respectively). As far as KNN is concerned, a difference of $1,000 in salary is enormous compared to a difference of 50 years in age. Consequently, salary will drive the KNN classification results, and age will have almost no effect. This is contrary to our intuition that a salary difference of $1000 is quite small compared to an age difference of 50 years. Furthermore, the importance of scale to the KNN classifier leads to another issue: if we measured salary in Japanese yen, or if we measured age in minutes, then we'd get quite different classification results from what we get if these two variables are measured in dollars and years.

A good way to handle this problem is to standardize the data so that all standardize variables are given a mean of zero and a standard deviation of one. Then all variables will be on a comparable scale. The scale() function does just scale() this. In standardizing the data, we exclude column 86, because that is the qualitative Purchase variable.

```
standardized.X=scale(Insurance_Data[,-86])
var(Insurance_Data[,1])
var(standardized.X[,1])
```

Now every column of `standardized.X` has a standard deviation of one and a mean of zero. We now split the observations into a test set, containing the first 1,000 observations, and a training set, containing the remaining observations. The vector `test` is numeric, with values from 1 through 1000. Typing `standardized.X[test,]` yields the submatrix of the data containing the observations whose indices range from 1 to 1000, whereas typing `standardized.X[-test,]` yields the submatrix containing the observations whose indices do not range from 1 to 1000.

```
test=1:1000
train.X=standardized.X[-test ,]
test.X=standardized.X[test ,]
train.Y=Purchase[-test]
test.Y=Purchase[test]
```

We fit a KNN model on the training data using K = 1, and evaluate its performance on the test data. It is a good practice to set seed for replicable results while training models.

```
#Set Seed
set.seed(1)
#Generate KNN model with K=1
knn.pred=knn(train.X,test.X,train.Y,k=1)
#Overall Error rate for K=1
mean(test.Y!=knn.pred)
#Error tate for always predict No
mean(test.Y!="No")
```

The KNN error rate on the 1,000 test observations is just under 12%. At first glance, this may appear to be fairly good. However, since only 6% of customers purchased insurance, we could get the error rate down to 6% by always predicting No regardless of the values of the predictors!

Suppose that there is some non-trivial cost to trying to sell insurance to a given individual. For instance, perhaps a salesperson must visit each potential customer. If the company tries to sell insurance to a random selection of customers, then the success rate will be only 6%, which may be far too low given the costs involved. Instead, the company would like to try to sell insurance only to customers who are likely to buy it. So the overall error rate is not of interest. Instead, the fraction of individuals that are correctly predicted to buy insurance is of interest.

Success Rate can be calculated in two ways: `mean(test.Y!=knn.pred)` gives the success rates as our response variable has 0s and 1s. It can also be calculated from the confusion matrix as `9/(68+9)`. It turns out that KNN with K = 1 does far better than random guessing among the customers that are predicted to buy insurance. Among 77 such customers, 9, or 11.7%, actually do purchase insurance. This is double the rate that one would obtain from random guessing (6%).

Suppose that we know the margin of each insurance policy sold is $1800 over 3 years, and cost of sale is $180. We can compute total profits if we had followed the marketing strategy suggested by KNN with K=1 by adding up total revenue from insurance policies sold and subtracting total cost of sales in the test dataset. In the example where K=1, this works out to 9*(1800)-77*(180)=$2340, which is greater than $0 from selling no policies.

```
# Confusion Matrix: Rate of Predicted vs Actual Test
# Values
table(knn.pred,test.Y)
9/(68+9)
```

```
> table(knn.pred,test.Y)
        test.Y
knn.pred  No Yes
     No  873  50
     Yes  68   9
```

Problem Set Questions 9.1

Run KNN on the dataset using 3 different values for K. Compute the success rate and profits for each approach. Discuss your results.

Comparison – Logistic Regression

As a comparison, we can also fit a logistic regression model to the data. If we use 0.5 as the predicted probability cut-off for the classifier, then we have a problem: only seven of the test observations are predicted to purchase insurance. Even worse, we are wrong about all of these! However, we are not required to use a cut-off of 0.5. If we instead predict a purchase any time the predicted probability of purchase exceeds 0. 25, we get much better results: we predict that 33 people will purchase insurance, and we are correct for about 33% of these people. This is over five times better than random guessing!

```
glm.fit=glm(Purchase ~ .,data=Insurance_Data,
family=binomial , subset =-test)
glm.probs=predict(glm.fit,Insurance_Data[test,],
type="response")

# Predictions with different probability cut-offs
glm.pred=rep("No",1000)
glm.pred[glm.probs>.5]="Yes"
table(glm.pred,test.Y)
glm.pred=rep("No",1000)
glm.pred[glm.probs>.25]="Yes"
table(glm.pred,test.Y)
```

4. Bibliography

1. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). An Introduction to Statistical Learning, Volume 112. Springer
2. Dataset Description - http://liacs.leidenuniv.nl/~puttenpwhvander/library/cc2000/data.html

5. Appendix

DATA DICTIONARY

Nr Name Description Domain

1 MOSTYPE Customer Subtype see L0
2 MAANTHUI Number of houses 1 – 10
3 MGEMOMV Avg size household 1 – 6
4 MGEMLEEF Avg age see L1
5 MOSHOOFD Customer main type see L2
6 MGODRK Roman catholic see L3
7 MGODPR Protestant ...
8 MGODOV Other religion
9 MGODGE No religion
10 MRELGE Married
11 MRELSA Living together
12 MRELOV Other relation
13 MFALLEEN Singles
14 MFGEKIND Household without children
15 MFWEKIND Household with children
16 MOPLHOOG High level education
17 MOPLMIDD Medium level education
18 MOPLLAAG Lower level education
19 MBERHOOG High status
20 MBERZELF Entrepreneur
21 MBERBOER Farmer
22 MBERMIDD Middle management
23 MBERARBG Skilled labourers
24 MBERARBO Unskilled labourers
25 MSKA Social class A
26 MSKB1 Social class B1
27 MSKB2 Social class B2
28 MSKC Social class C
29 MSKD Social class D
30 MHHUUR Rented house
31 MHKOOP Home owners
32 MAUT1 1 car
33 MAUT2 2 cars
34 MAUT0 No car
35 MZFONDS National Health Service
36 MZPART Private health insurance
37 MINKM30 Income < 30.000
38 MINK3045 Income 30-45.000
39 MINK4575 Income 45-75.000
40 MINK7512 Income 75-122.000
41 MINK123M Income >123.000
42 MINKGEM Average income
43 MKOOPKLA Purchasing power class
44 PWAPART Contribution private third party insurance see L4

45 PWABEDR Contribution third party insurance (firms) ...
46 PWALAND Contribution third party insurane (agriculture)
47 PPERSAUT Contribution car policies
48 PBESAUT Contribution delivery van policies
49 PMOTSCO Contribution motorcycle/scooter policies
50 PVRAAUT Contribution lorry policies
51 PAANHANG Contribution trailer policies
52 PTRACTOR Contribution tractor policies
53 PWERKT Contribution agricultural machines policies
54 PBROM Contribution moped policies
55 PLEVEN Contribution life insurances
56 PPERSONG Contribution private accident insurance policies
57 PGEZONG Contribution family accidents insurance policies
58 PWAOREG Contribution disability insurance policies
59 PBRAND Contribution fire policies
60 PZEILPL Contribution surfboard policies
61 PPLEZIER Contribution boat policies
62 PFIETS Contribution bicycle policies
63 PINBOED Contribution property insurance policies
64 PBYSTAND Contribution social security insurance policies
65 AWAPART Number of private third party insurance 1 - 12
66 AWABEDR Number of third party insurance (firms) ...
67 AWALAND Number of third party insurane (agriculture)
68 APERSAUT Number of car policies
69 ABESAUT Number of delivery van policies
70 AMOTSCO Number of motorcycle/scooter policies
71 AVRAAUT Number of lorry policies
72 AAANHANG Number of trailer policies
73 ATRACTOR Number of tractor policies
74 AWERKT Number of agricultural machines policies
75 ABROM Number of moped policies
76 ALEVEN Number of life insurances
77 APERSONG Number of private accident insurance policies
78 AGEZONG Number of family accidents insurance policies
79 AWAOREG Number of disability insurance policies
80 ABRAND Number of fire policies
81 AZEILPL Number of surfboard policies
82 APLEZIER Number of boat policies
83 AFIETS Number of bicycle policies
84 AINBOED Number of property insurance policies
85 ABYSTAND Number of social security insurance policies
86 AUTO Number of auto insurance policies 0 - 1

L0:
Value Label
1 1 High Income, expensive child
2 2 Very Important Provincials
3 3 High status seniors
4 4 Affluent senior apartments
5 5 Mixed seniors
6 6 Career and childcare

7 7 Dinki's (double income no kids)
8 8 Middle class families
9 9 Modern, complete families
10 10 Stable family
11 11 Family starters
12 12 Affluent young families
13 13 Young all american family
14 14 Junior cosmopolitan
15 15 Senior cosmopolitans
16 16 Students in apartments
17 17 Fresh masters in the city
18 18 Single youth
19 19 Suburban youth
20 20 Etnically diverse
21 21 Young urban have-nots
22 22 Mixed apartment dwellers
23 23 Young and rising
24 24 Young, low educated
25 25 Young seniors in the city
26 26 Own home elderly
27 27 Seniors in apartments
28 28 Residential elderly
29 29 Porchless seniors: no front yard
30 30 Religious elderly singles
31 31 Low income catholics
32 32 Mixed seniors
33 33 Lower class large families
34 34 Large family, employed child
35 35 Village families
36 36 Couples with teens 'Married with children'
37 37 Mixed small town dwellers
38 38 Traditional families
39 39 Large religous families
40 40 Large family farms
41 41 Mixed rurals

L1:
1 20-30 years
2 30-40 years
3 40-50 years
4 50-60 years
5 60-70 years
6 70-80 years

L2:
1 Successful hedonists
2 Driven Growers
3 Average Family
4 Career Loners
5 Living well

6 Cruising Seniors
7 Retired and Religeous
8 Family with grown ups
9 Conservative families
10 Farmers

L3:
0 0%
1 1 - 10%
2 11 - 23%
3 24 - 36%
4 37 - 49%
5 50 - 62%
6 63 - 75%
7 76 - 88%
8 89 - 99%
9 100%

L4:
0 f 0
1 f 1 – 49
2 f 50 – 99
3 f 100 – 199
4 f 200 – 499
5 f 500 – 999
6 f 1000 – 4999
7 f 5000 – 9999
8 f 10.000 - 19.999
9 f 20.000 - ?